Simulazione di Agenti BDI basati su Prolog in Alchemist

Tesi in: Sistemi Autonomi

Relatore:
Chiar.mo Prof.
Andrea Omicini

Presentata da:
Filippo Nicolini

Correlatori:
Dott. Ing. Danilo Pianini
Dott. Giovanni Ciatto

Alma Mater Studiorum – Università di Bologna
Campus di Cesena

12 Dicembre 2019

Contesto

Oggi

Nel mondo ad agenti presenti ambienti con programmazione espressiva oppure con orientamento alle prestazioni delle simulazioni.

Jason

Interprete di versione estesa di AgentSpeak che migliora abilità degli agenti e semplifica il linguaggio.

Multi-Agent Research and Simulation – MARS

Framework per simulazioni distribuite di agenti programmati ad alto livello.

Obiettivo della tesi

Obiettivo

Unificare piattaforme orientate alla programmazione di agenti con ambienti di simulazione di agenti.

Ambiente

Per unificare programmazione e simulazione si è voluto portare il modello di agenti BDI all'interno di Alchemist.

Interprete

Per portare il modello BDI in Alchemist si è scelto di utilizzare tuProlog per creare un interprete multi-paradigma.

Scelte tecnologiche

Motivazione

Alchemist e tuProlog sono solidi e supportati da gruppi in modo attivo che semplificano la manutenibilità del progetto.

Alchemist

Meta-modello flessibile per implementare modelli diversi e potenza di calcolo per eseguire simulazioni.

tuProlog

È multi-paradigma, ovvero permette di integrare un interprete Prolog con piattaforme/linguaggi OO.

AgentSpeak, modello BDI, tuProlog

Modello BDI

Modello BDI (Beliefs, Desires, Intentions) implementa gli aspetti principali del ragionamento umano per programmare agenti intelligenti.

AgentSpeak

Linguaggio orientato agli agenti basato su modello BDI e programmazione logica.

tuProlog

Libreria che permette di utilizzare Prolog all'interno di applicazioni e infrastrutture distribuite sfruttando un core minimale.

Interprete tuProlog di AgentSpeak

Formalizzazione

Estenzione di AgentSpeak realizzando un'interprete attraverso definizione di alcune sintassi.

API – agente

- inizializzazione agente
- invocazioni verso linguaggio OO
- gestione 'belief base'
- gestione eventi e posizionamento

API – interprete

- verifica contesto e recupero corpo del piano
- esecuzione intenzione, azioni o goal

Alchemist

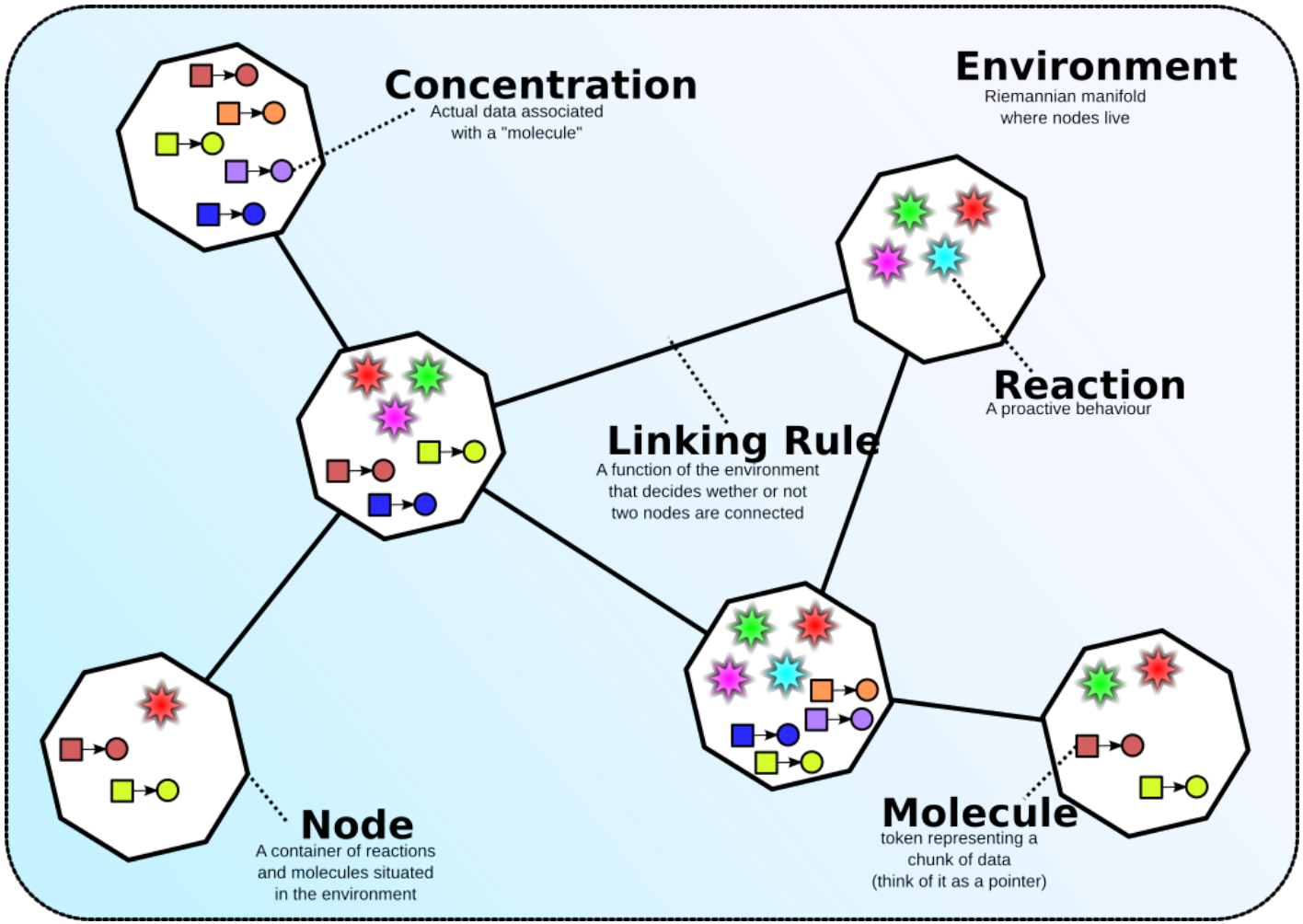
Alchemist

Simulatore per il calcolo pervasivo, aggregato e naturale che si basa su un meta-modello flessibile e che permette implementazioni di modelli diversi tra loro.

Meta-modello

- Node
- Environment
- Reaction (Time Distribution, Condition, Action)
- Linking Rule

Meta-modello



Spazi di tuple

LINDA

Modello di coordinazione e comunicazione tra processi paralleli con memoria associativa, virtuale, condivisa.

Spatial Tuples

Estensione del modello base di tuple per i sistemi distribuiti multi-agente.

Operazioni consentite

- in: legge tupla e la consuma
- rd: legge tupla senza consumarla
- out: inserisce tupla

Ciclo di ragionamento

Percezioni

Informazioni ricevute tramite un apparato con le quali l'agente percepisce i cambiamenti dell'ambiente.

Eventi

Sono relativi a percezioni che l'agente ha ricevuto e possono essere catturati dall'agente.

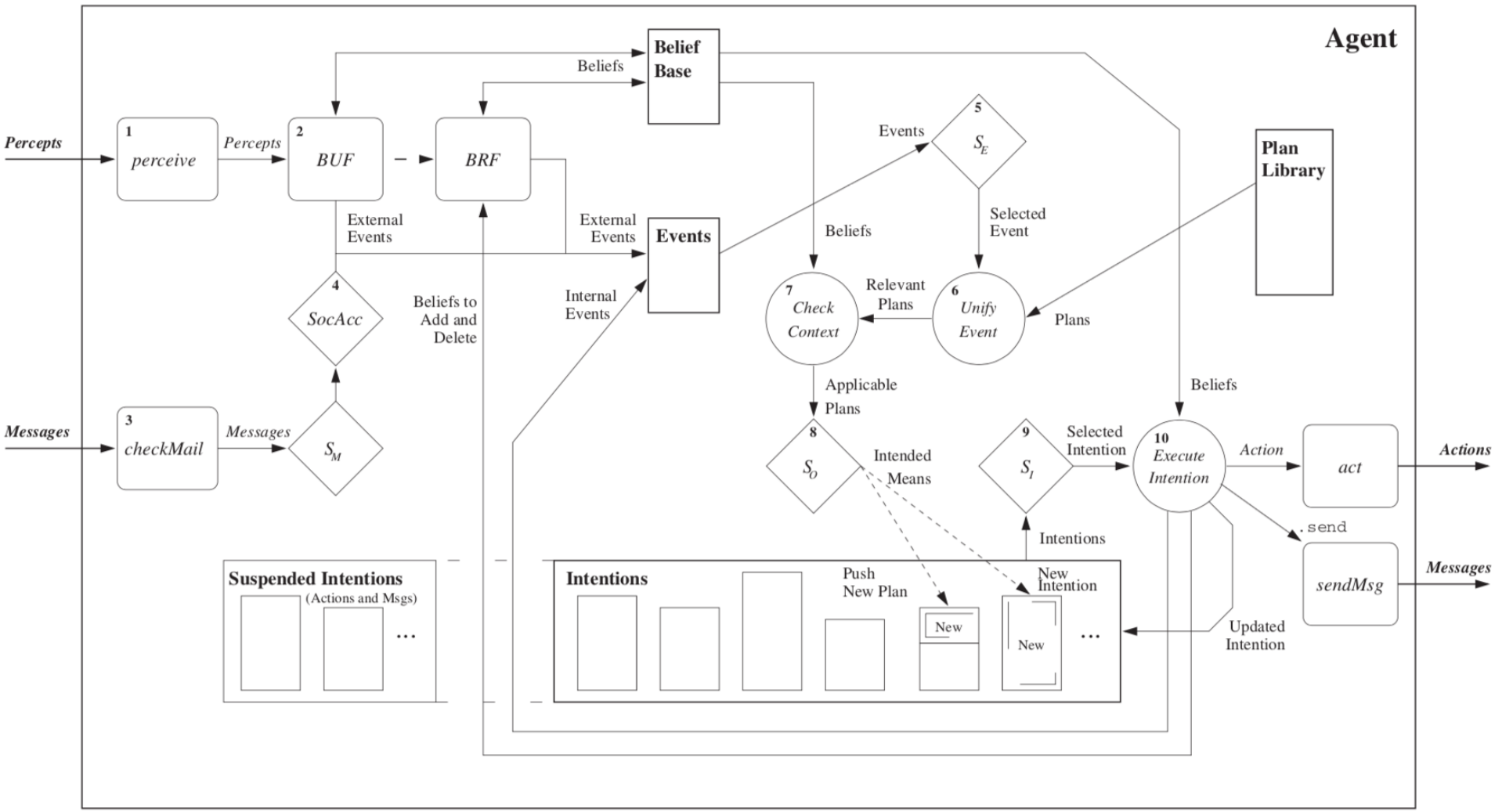
Piani

Definiscono come l'agente agisce per raggiungere goal.

Intenzioni

Operazioni che l'agente vuole eseguire per portare a termine un certo goal.

Ciclo di ragionamento

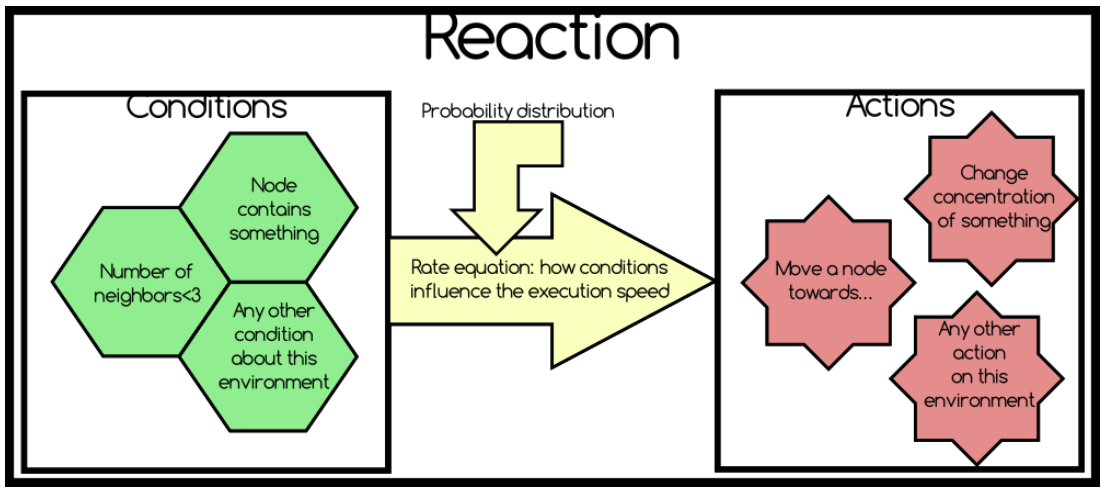


Unione modelli

Mapping

Environment → Spazio agenti Nodo → Contenitore agenti

Reazione → Agente



Caratteristiche mapping

Ricercata la massima espressività lavorando su più strati: nodo può essere inteso come device in cui operano più agenti.

Scenario

Goldminers

Un gruppo di minatori deve recuperare pepite d'oro da miniere sparse nell'ambiente e riportarle in un deposito.

Entità → ruoli

All'interno del problema si individuano le seguenti entità:

- minatori → agenti
- pepite → tuple
- miniere → spazi di tuple
- deposito → agente

Realizzazione

Minatore

Comportamento diviso in 4 stati:

- ricerca: spostamento casuale emettendo richieste di tuple;
- ricezione tupla: salva posizione miniera e si dirige al deposito;
- arrivo deposito: invio pepita e si dirige posizione miniera salvata;
- arrivo miniera: torna stato ricerca.

Miniera

Istanzia N tuple all'inizializzazione e risponde alle richieste dei minatori.

Deposito

Statico nell'ambiente, riceve la pepita tramite un messaggio.

Conclusioni e lavori futuri

Conclusioni

- Realizzazione di interprete per programmare e simulare agenti
- Forte espressività modello e interprete (Prolog e OO)
- Estensione agenti in spazi di tuple

Lavori futuri

- Implementazione interprete OO in piattaforme per ambiente reale
- Ricerca per migliorie sia nell'interprete Prolog che nell'interprete OO

Simulazione di Agenti BDI basati su Prolog in Alchemist

Tesi in: Sistemi Autonomi

Relatore:
Chiar.mo Prof.
Andrea Omicini

Presentata da:
Filippo Nicolini

Correlatori:
Dott. Ing. Danilo Pianini
Dott. Giovanni Ciatto

Alma Mater Studiorum – Università di Bologna
Campus di Cesena

12 Dicembre 2019