Introduzione

Questo lavoro ha l'obiettivo di portare sul simulatore Alchemist il modello ad agenti.

Alchemist è un meta-simulatore estendibile, ispirato alla chimica stocastica e adatto al calcolo pervasivo e ai sistemi distribuiti. Fornisce un meta-modello flessibile, sul quale gli sviluppatori legano le proprie astrazioni, realizzando una cosiddetta 'incarnazione'.

Il modello ad agenti a cui si fa riferimento è quello BDI (Beliefs, Desires, Intentions) che è ispirato al modello del comportamento umano.

Indice

Elenco delle figure

Capitolo 1

Alchemist

Alchemist fornisce un ambiente di simulazione sul quale è possibile sviluppare nuove incarnazioni, ovvero nuove definizioni di modelli da sviluppare su di esso.

1.1 Il meta-modello

Il meta-modello di Alchemist può essere compreso con la figura 1.1.

L'***Environment*** è l'astrazione dello spazio ed è anche l'entità più esterna che funge da contenitore per i *nodi*. Conosce la posizione di ogni *nodo* nello spazio ed è quindi in grado di fornire la distanza tra due di essi e ne permette inoltre lo spostamento.

È detta ***Linking rule*** una funzione dello stato corrente dell'*environment* che associa ad ogni *nodo* un ***Vicinato***, il quale è un entità composta dal *nodo* e da un set di *nodi*, i vicini.

Un ***Nodo*** è un contenitore di *molecole* e *reazioni* che è posizinato all'interno di un *environment*.

La ***Molecola*** è il nome di un dato, paragonabile a quello che rappresenta il nome di una variabile per i linguaggi imperativi. Il valore da associare ad una *molecola* è detto ***Concentrazione***.

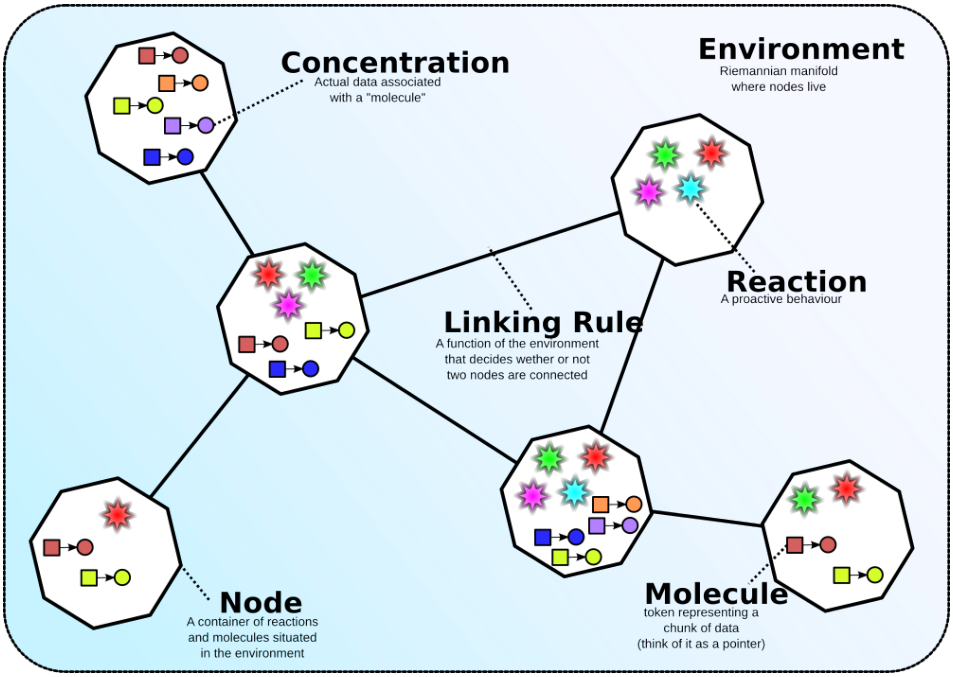

Figura 1.1: Illustrazione meta-modello di Alchemist

Una ***Reazione*** è un qualsiasi evento che può cambiare lo stato dell'*environment* ed è. definita tramite una lista di *condizioni*, una o più *azioni* e una *distribuzione temporale*.
La frequenza con cui avvengono dipende da:

- un parametro statico di frequenza;
- il valore di ogni condizione;
- un'equazione di frequenza che combina il tasso statico e il valore delle condizioni restituendo la frequenza istantanea;
- una distribuzione temporale.

Ogni *nodo* contiene un set di *reazioni* che può essere anche vuoto.

Per comprendere meglio il meccanismo di una reazione si può osservare la figura 1.2.

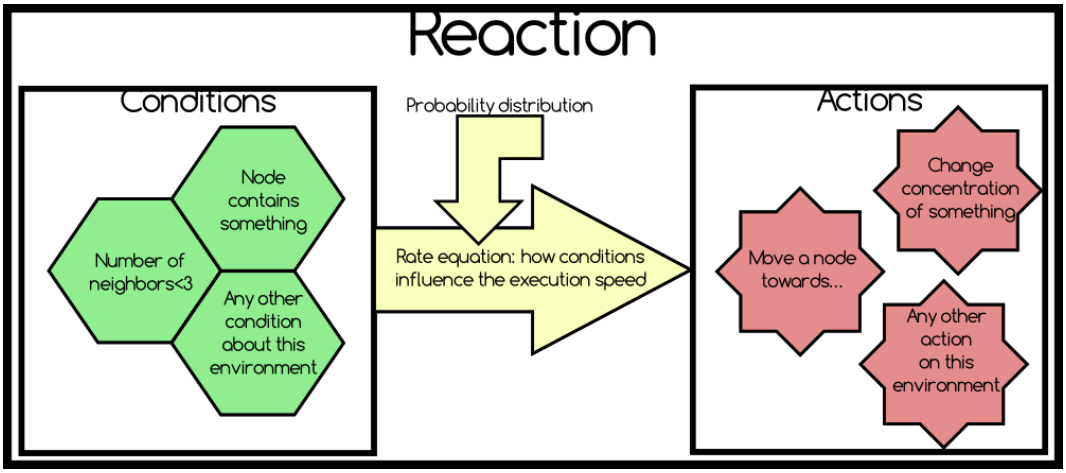

Figura 1.2: Illustrazione modello reazione di Alchemist

Una ***Condizione*** è una funzione che prende come input l'*environment* corrente e restituisce come output un booleano e un numero. Se la *condizione* non si verifica, le *azioni* associate a quella *reazione* non saranno eseguite. In relazione a parametri di configurazione e alla *distribuzione temporale*, una *condizione* potrebbe influire sulla velocità della *reazione*.

Un'***Azione*** è la definizione di una serie di operazioni che modellano un cambiamento nell'*environment*.

In Alchemist un'incarnazione è un'istanza concreta del meta-modello appena descritta e che implementa una serie di componenti base come: la definizione di una molecola e del tipo di dati della concentrazione, un set di condizioni, le azioni e le reazioni. Incarnazioni diverse possono modellare universi completamente differenti.

1.2 Scrivere una simulazione

Capitolo 2

Agenti

Un'agente è un entità che agisce in modo autonomo e continuo in uno spazio condiviso con altri agenti. Le caratteristiche principali di un agente sono: autonomia, proattività e reattività. Gli agenti sono formati da un nome, che è una caratteristica statica, e da componenti dinamici come lo stato.

2.1 Agenti BDI

Gli agenti BDI forniscono un meccanismo per separare le attività di selezione di un piano, presente nella lista dei piani, dall'esecuzione del piano attivo e quindi permettono di bilanciare il tempo speso nella scelta del piano e quello per eseguirlo.

I ***Beliefs*** sono informazioni dello stato dell'agente, ovvero quello che lui sa del mondo (di se stesso e degli altri agenti), e possono comprendere regole di inferenza per permettere l'aggiunta di nuovi *belief*.
Inoltre si possono modificare nel tempo: l'insieme dei *belief* di un agente è dettto 'belief base' o 'belief set'.

I ***Desires*** sono tutti i possibili piani che l'agente potrebbe eseguire. Rappresentano gli obiettivi o le situazioni che l'agente vorrebbe realizzare o portare a termine. I *goals* sono *desires* che l'agente persegue attivamente: per

questo motivo, in generale, i piani desiderabili possono non essere coerenti tra loro mentre i *goals* è bene che lo siano.

I ***Intentions*** sono piani a cui l'agente ha deciso di lavorare o a cui sta già lavorando. I piani sono sequenze di azioni che un agente può eseguire per raggiungere un'*intention* e a loro volta ne possono includere altri.

Gli ***Eventi*** innescano le attività reattive degli agenti il cui risultato può essere l'aggiornamento dei *beliefs*, la chiamata ad altri piani o la modifica di *goals*.

Il ciclo di ragionamento di un agente avviene come descritto in figura 2.1.

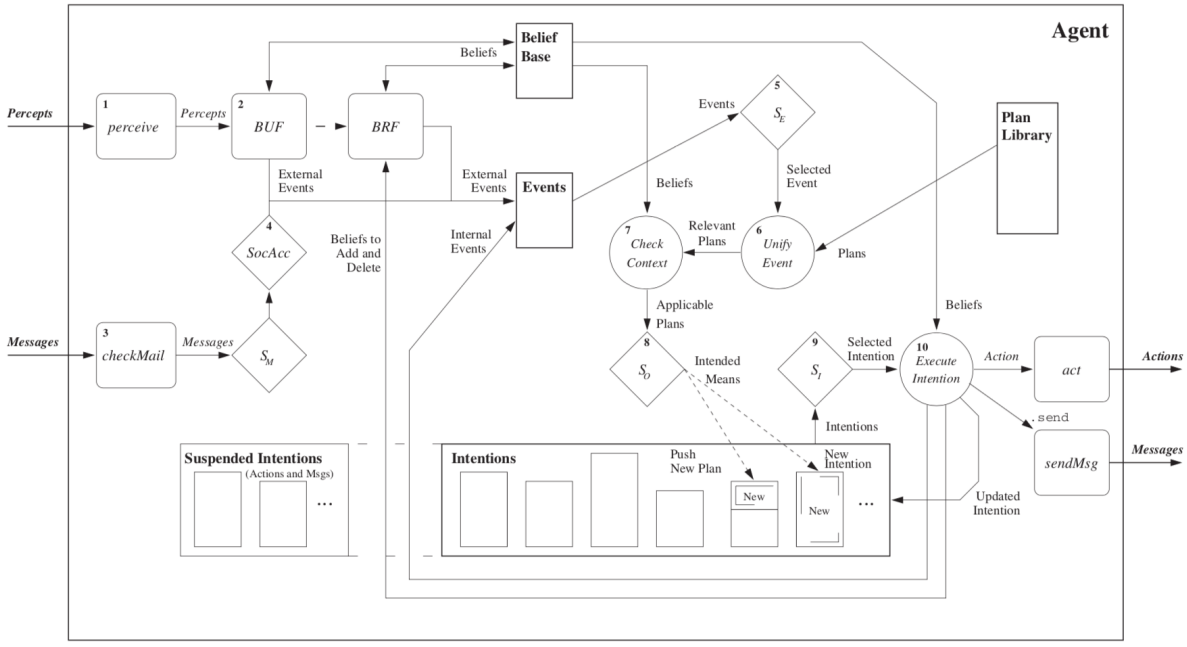

Figura 2.1: Ciclo di ragionamento di un agente

2.2 Ciclo di ragionamento

RIVEDERE IL PEZZO QUI SOTTO E CONTINUARLO....

La prima cosa che fa un agente nel ciclo di ragionamento è controllare l'ambiente per aggiornare i *beliefs* con lo stato attuale. Quindi è necessario che l'agente abbia dei componenti per percepire i cambiamenti dell'ambiente. Una volta ottenuta la lista dei cambiamenti percepiti è necessario aggiornare la *belief base* (che contiene l'insieme dei *beliefs* dell'utente allo stato attuale).

// TODO CONTINUARE la descrizione del ciclo di ragionamento dell'agente....

Capitolo 3

Progetto

Il progetto, come descritto nell'introduzione, ha come obiettivo l'implementazione del meta-modello di Alchemist attraverso la definizione di un incarnazione che modelli gli agenti all'interno del simulatore.

3.1 Mapping dei modelli

Il primo passo nell'evoluzione del progetto è stata l'analisi del mapping tra i due modelli, necessaria per individuare eventuali incongruenze o evidenziare opportunità a livello applicativo, maggiore espressività. Nei mapping effettuati si è cercato quindi di individuare l'entità del meta-modello di Alchemist che offrisse maggiori opportunità espressive per la definizione dell'agente.

Nella prima prova, l'agente è stato rapportato ad un nodo da cui ne deriva che l'environment sarà lo spazio che conterrà tutti gli agenti mentre, internamente al nodo, le molecole e le concentrazioni saranno utilizzate per gestire i beliefs dell'agente e le reazioni che saranno riferite ai piani utilizzando le condizioni come clausola per far scattare le azioni.
Questo tipo di mapping consentirebbe di realizzare simulazioni di sistemi non complessi in cui vi è un solo 'livello' di agenti che interagiscono tra loro. Questa affermazione può essere compresa meglio analizzando il secondo tentativo che è stato effettuato.

Nel secondo mapping, l'agente è stato spostato più internamente al nodo, in modo da far diventare il nodo stesso uno spazio per gli agenti che sono quindi mappati come reazioni. In questo modo l'environment sarà uno spazio in cui possono essere presenti più nodi, i quali a loro volta potranno contenere uno o più agenti. La frequenza con cui gli eventi sono innescati dipende, oltre che dai parametri passati nella configurazione della simulazione, anche dalle condizioni definite per quello specifico agente.
In questo caso si riuscirà a creare un sistema con più agenti all'interno di un singolo nodo, che in ambito applicativo può essere riferito ad un device, e che si muoverà nello spazio insieme ad altri nodi, che conterranno altri agenti.

3.2 Fasi di sviluppo

Il passo successivo è stato quello di stilare un piano di sviluppo per affrontare il problema attraverso step incrementali. Questa scelta è stata dettata anche dalla poca conoscenza iniziale per lo sviluppo in Alchemist. Le fasi in cui è stato suddiviso il progetto sono state le seguenti:

1. Esecuzione di un'istruzione elementare da parte di un agente (ad esempio contatore);

2. Scambio di messaggi tra due agenti (ad esempio ping pong): inizialmente gli agenti possono anche risiedere nello stesso nodo ma poi l'implementazione deve prevedere il posizionamento degli agenti su nodi diversi;

3. Gestione del flusso di controllo attraverso l'inserimento di un operazione (come lo spostamento del nodo o l'incremento di un contatore) prima di effettuare la risposta al messaggio;

4. Gestione del flusso condizionato inserendo una clausola per la quale lo scambio di messaggi debba avvenire o meno.

3.3 Implementazioni

Dopo aver esaminato i due modelli e aver analizzato i mapping realizzati si è deciso di implementare il caso che mappa l'agente alla reazione poichè seguendo lo schema del meta-modello di Alchemist l'implementazione risulta più immediata e espressiva.

3.3.1 Scambio di messaggi tra agenti

Lo sviluppo è partito dalla definizione della classe **AgentIncarnation** che implementa l'interfaccia Incarnation. I metodi definiti nell'interfaccia consentono di caratterizzare l'incarnazione nella creazione delle varie entità del modello (nodi, distribuzioni temporali, reazioni, condizioni, azioni).

Per la creazione del nodo si è definita la classe **AgentsContainerNode** che estende AbstractNode. Questa classe ha tra le sue proprietà il riferiemnto all'environment in cui si trova il nodo e una struttura dati composta da coppie chiave e valore in cui la chiave è il nome dell'agente e il valore è il riferimento all'azione dell'agente.

La distribuzione temporale di ogni reazione è stata realizzata istanziando la classe **DiracComb** inizializzata con il parametro recuperato dal file di configurazione della simulazione. La classe permette di emettere eventi ad un intervallo temporale specificato dal parametro passato.

Per le reazioni è stata definita la classe **AgentReaction** che rappresenta l'agente e che contiene le condizioni che devono verificarsi per far avvenire le azioni associate ad essa. Come proprietà della classe è presente solo una stringa che memorizza il nome dell'agente.
La creazione delle condizioni, per la fase iniziale del progetto, è stata fatta istanziando la classe **AbstractCondition** e implementando i metodi rimasti dell'interfaccia Condition: *getContext* (definisce la profondità della condizione GLOBAL,LOCAL,NEIGHBORHOOD), *getPropensityContribution* (permette di influenzare la velocitÃ della reazione che decide se utilizzare o meno questo parametro), *isValid* (definisce la clausola per la valità della condizio-

ne)).

L'azione da creare, ovvero il ciclo di ragionamento dell'agente, è passata della reazione come parametro. Per la prima fase, che prevedee un semplice scambio di messaggi, sono state definite le classi **SimpleAgentAction** e **PostmanAction** che estende la precedente. La prima ha come proprietÃ il nome dell'agente, un flag per l'inizializzazione, una mailbox formata da due code (una per la posta in entrata e una per quella in uscita) e un motore tuProlog. Al suo interno sono implementati i metodi *execute* (che è il metodo principale in cui avviene il ciclo di ragionamento) e i metodi per la gestione delle caselle dei messaggi, le cui strutture sono definite come classi innestate InMessage e OutMessage rispettivamente per i messagggi in entrata e in uscita.

La classe **PostmanAction** sovrascrive l'implementazione del metodo *execute* per invocare, ad ogni evento invocato dal simulatore, un metodo nel nodo che provvederà a prelevare i messaggi in uscita da ogni agente e recapitarli ai corretti destinatari.

Bibliografia

[1] alchemistsimulator.github.io

[2] Programming Multi-Agent Systems in AgentSpeak using Jason, (Rafael H. Bordini, Jomi Fred Hübner, Michael Wooldridge), Wiley, Interscience (2007)