Corso di laurea magistrale in Informatica

**Corso di Intelligenza Artificiale e Laboratorio**

**Docente modulo : P.Torasso**

**a.a. 2013/2014**

**Progetto: NG\_CAFE**

Sommario

[IL DOMINIO 1](#_Toc378948209)

[LE FUNZIONALITA’ DELL’AGENTE ROBOTICO 3](#_Toc378948210)

[LE AZIONI 3](#_Toc378948211)

[CONOSCENZA A PRIORI SULL’AMBIENTE 6](#_Toc378948212)

[I MESSAGGI 6](#_Toc378948213)

[LE PERCEZIONI 7](#_Toc378948214)

[IL TASK E LA VALUTAZIONE DELL’UTILITA’ 8](#_Toc378948215)

[L’IMPLEMENTAZIONE 9](#_Toc378948216)

[MAIN 10](#_Toc378948217)

[ENV 12](#_Toc378948218)

# 

# IL DOMINIO

Il dominio di riferimento (estremamente semplificato) prende lo spunto da problematiche di robotica mobile di servizio (service robotics) in un contesto applicativo di supporto al lavoro presso NG-CAFE (next generation cafe). In particolare si intende sviluppare un agente che svolga il ruolo di un cameriere che serve ai tavoli di NG Cafe. Nel progetto si prendono in considerazione solo alcune attività:

* Acquisire le ordinazioni da parte dei clienti di un tavolo (compito facilitato perché presso NG cafe, gli utenti ordinano tramite appositi smart device e quindi l’agente riceve direttamente avviso da parte del sistema di ordinazione)
* Portare al tavolo quanto ordinato dai clienti di quel tavolo
* Sparecchiare il tavolo, una volta che i clienti hanno finito la consumazione
* Riporre i rifiuti in appositi contenitori.

L’agente artificiale incaricato del compito è un robot mobile che deve essere in grado di svolgere il compito in modo autonomo essendo dotato di:

* Capacità motorie che permettono di muoversi nell’ambiente
* capacità di manipolazione che gli permettono di svolgere certi tipi di azione
* capacità sensoriali che permettono di analizzare l’ambiente circostante con un certo grado di dettaglio.
* Capacità deliberative che permettono all’agente di decidere quali azioni compiere e quando compierle

Il progetto in questione ha lo scopo di far sviluppare un agente intelligente che combina sia aspetti deliberativi che reattivi per guidare l’attività del robot fisico. Visto il contesto in cui il progetto didattico è inserito sono state fatte due assunzioni di base (che nel mondo reale sono assai problematiche):

* le azioni che l’agente decide di compiere sono sempre eseguite con successo (non si modella la possibilità che le azioni possano fallire se le azioni vengono eseguite in un contesto in le precondizioni per l’applicabilità dell’azione sono verificate)
* Per quanto riguarda la parte percettiva si suppone che il robot sia dotato di sensori e di processi interpretativi che restituiscono la situazione effettiva del mondo senza alcun errore o ambiguità. Tuttavia si considera che i sensori abbiano una copertura limitata, per cui il mondo risulta osservabile solo localmente.

L’ambiente in cui opera il robot è modellato in modo molto semplificato ed astratto. Ciò nonostante presenta alcune caratteristiche che lo rendono non banale. Infatti, l’ambiente in cui deve operare il robot è un (tipico) Cafè dove sono presenti:

* Clienti che posso muoversi nel caffè oppure stare seduti ad un tavolino per la consumazione
* Tavolini che hanno associato un numero prefissato di posti a sedere (nella versione corrente max 4)
* Dei dispenser dove l’agente può andare a prelevare cibo (FoodDispenser) o bevande (DrinkDispenser).
* i TrashBasket dove l’agente va a depositare i resti della consumazione (food)
* RecyclableBasket dove l’agente va a depositare i contenitori utilizzati per servire le bevende.

Per ragioni di semplicità, la rappresentazione dell’ambiente è effettuata per mezzo di una griglia composte da celle (connesse tra di loro) di forma quadrata e della stessa dimensione. Il sistema di sensori visivi presenti a bordo del robot è sempre in grado di classificare la natura delle celle circostanti la posizione del robot stesso. In particolare ciascuna cella può essere di uno solo dei seguenti tipi:

* Wall: rappresenta una parete
* Empty: è una cella libera per cui il robot può occuparla e/o attraversarla, cosi come un cliente. Ovviamente la cella quando è occupata dal robot o da un cliente non è più empty.
* Table. Per semplicità ogni table occupa una sola cella. Si suppone inoltre che la table abbia spazio sufficiente per contenere le consumazioni ordinate solo dai clienti dello specifico Table. Prima di poter far accomodare altri ad un tavolo è necessario che il table sia in stato clean (tutte i resti di consumazione sono stati asportati)
* Seat: occupa una sola cella e può ospitare al massimo una persona. Si noti che è in una cella attigua al table corrispondente. Come detto per ogni table ci sono al max 4 seat
* TrashBasket è un contenitore che occupa una sola cella ed è destinato a riporre gli avanzi della consumazione. Per ragioni di semplicità si assume che il Trash basket abbia capacità infinita
* RecyclableBasket è un contenitore che occupa una sola cella e dove l’agente va a depositare i contenitori (tazze, bottigliette e/o lattine) utilizzate per servire le consumazioni. Anche in questo caso si assume che abbia capacità infinita
* DrinkDispenser è un contenitore che occupa una sola cella e dove il robot può rifornirsi di bevande. Per semplicità si assume che la sua capacità sia infinita
* FoodDispenser. è un contenitore che occupa una sola cella e dove il robot può rifornirsi di cibo (panini/pizzette, ecc.). Per semplicità si assume che la sua capacità sia infinita.
* Person. Ogni persona occupa un seat se è seduto ad un tavolo, altrimenti può occupare una cella “empty” per attraversare l’ambiente. Il robot riconosce la presenza di una persona, ma non ha capacità di individuare quale persona sia.
* Parking: è la cella in cui il robot parcheggia e dove può ricaricare le batterie.

Si assume che:

* l’ambiente sia completamente racchiuso da pareti/porte (per semplicità modellati con un unico tipo wall),
* il robot possa muoversi solo in celle di tipo empty o nel parking. Il tentativo del robot di entrare in una cella occupata non ha successo e il robot rimane nella cella di partenza (con qualche conseguenza sulla valutazione delle capacità dell’agente robotico, si veda nel seguito).
* le persone possano muoversi solo in celle libere (oltre che ad essere sedute in seat).

# LE FUNZIONALITA’ DELL’AGENTE ROBOTICO

Come accennato sopra, l’agente robotico è un robot di servizio dotato di opportune funzionalità tra cui quelle locomotorie (mobilità), sensoriali (vedi sotto) e di manipolazione (capacità di movimentare oggetti – drink e food).

Il robot può muoversi in avanti in una delle quattro direzioni (north, south, west, east) e può cambiare direzione con una apposita manovra ruotando di 90 gradi a destra o a sinistra. Il robot non può andare ad occupare una cella che sia occupata da qualche altra entità.

Il robot inoltre è in grado (grazie alla capacità di manipolazione) di caricare a bordo del robot stesso fino a quattro oggetti di tipo food e/o drink.

Ovviamente l’operazione di carico può avvenire solo in luoghi deputati, in particolare il robot può caricare gli oggetti solo nell’apposito dispenser. Il robot è in grado di consegnare gli ordini sul tavolo da cui provengono le ordinazioni.

Il robot ha anche il compito di ripulire il tavolo dopo che i clienti hanno terminato le consumazioni e deve riportare al TrashBasket e/o al RecyclableBasket i resti delle consumazioni, dove il robot è in grado di svuotarli.

Si noti che:

* Il robot è autorizzato a trasportare insieme drink e food per tavoli diversi (se rispetta i limiti di capacità di carico).
* Per costruzione il robot non può trasportare contemporaneamente consumazioni e rifiuti (le azioni falliscono, si veda sotto). Pertanto, per caricare i rifiuti di un tavolo il robot deve essere scarico
* Il robot ha spazio sufficiente (in due comparti distinti) per caricare i contenitori di bevande consumate e resti del cibo (anche di più tavoli)

## LE AZIONI

Ad ogni istante di tempo il robot può fare una sola delle azioni sotto elencate. Si noti che le azioni hanno una loro durata e non sono interrompibili. Per cui l’agente può iniziare la prossima azione solo quando la precedente si è conclusa (con successo e con fallimento).

**Forward**

Con questa azione il robot avanza di una posizione (cella) nella direzione in cui è attualmente orientato. L’azione ha successo se la cella in cui si dirige è empty (oppure parking). Nel caso il robot cerchi di andare in una cella non libera, l’azione fallisce nel senso che il robot rimane posizionato dove è, ma subisce un qualche danno (più sotto si misurano le prestazioni del robot in termini di penalty).

Questa azione richiede 2 unità di tempo per essere eseguita.

**Turnright**

Con questa azione il robot ruota su se stesso di 90 gradi a destra rispetto alla sua direzione corrente. Si tenga presente che le direzioni sono 4 (in particolare : north, south, east, west). Il robot non cambia cella con questa operazione. Questa azione non fallisce mai.

Questa azione richiede 1 unità di tempo per essere eseguita.

**Turnleft**

Con questa azione robot ruota su se stesso di 90 gradi a sinistra rispetto alla sua direzione corrente. Le condizioni di applicazioni sono esattamente uguali a quelle di Turnright

Questa azione richiede 1 unità di tempo per essere eseguita.

**LoadDrink(x,y)**.

Con questa azione il robot carica a bordo una bevanda prelevandolo dal DrinkDispenser situato in posizione (x,y). Per poter eseguire l’azione, il robot deve essere in una delle celle accessibili adiacenti (secondo le 4 direzioni) rispetto al dispenser. Si noti che l’azione fallisce se il robot è già carico (tutte le posizione sono piene), o l’agente robotico non è nella cella giusta per fare operazione di load, oppure se la cella (x,y) non contiene un DrinkDispenser.

Questa azione richiede 6 unità di tempo per essere eseguita.

**LoadFood(x,y)**.

Con questa azione il robot carica a bordo un cibo (una porzione di cibo) prelevandolo dal FoodDispenser situato in posizione (x,y). Per poter eseguire l’azione, il robot deve essere in una delle celle accessibili adiacenti (secondo le 4 direzioni) rispetto al dispenser. Si noti che l’azione fallisce se il robot è già carico (tutte le posizione sono piene), o l’agente robotico non è nella cella giusta per fare operazione di load, oppure se la cella (x,y) non contiene un FoodDispenser.

Questa azione richiede 5 unità di tempo per essere eseguita.

**DeliveryFood(x,y)**

Con questa azione il robot deposita una porzione di cibo (che è a bordo) sul tavolo che si trova nella cella (x,y). L’azione per poter essere eseguita richiede che il robot sia in una delle quattro celle adiacenti (secondo le 4 direzioni) rispetto alla cella (x,y) in cui si trova il tavolo. Si noti che l’azione fallisce se la cella (x,y) non contiene un tavolo, se il cibo non è a bordo del robot, se il robot non è in una posizione contigua al tavolo.

Questa azione richiede 4 unità di tempo per essere eseguita.

**DeliveryDrink(x,y)**

Con questa azione il robot deposita una bevanda (che è a bordo) sul tavolo che si trova nella cella (x,y). L’azione per poter essere eseguita richiede che il robot sia in una delle quattro celle adiacenti (secondo le 4 direzioni) rispetto alla cella (x,y) in cui si trova tavolo. Si noti che l’azione fallisce se la cella (x,y) non contiene un tavolo, se la bevanda non è a bordo del robot, se il robot non è in una posizione contigua al tavolo.

Questa azione richiede 4 unità di tempo per essere eseguita.

**CleanTable(x,y)**

Con questa azione il robot ripulisce il tavolo che si trova nella cella (x,y) dagli avanzi della consumazione di cibo e dai contenitori (vuoti) delle bevande. L’azione per poter essere eseguita richiede che il robot sia scarico e che sia in una delle quattro celle adiacenti (secondo le 4 direzioni) rispetto alla cella (x,y) in cui si trova il tavolo. Pertanto l’azione fallisce se la cella (x,y) non contiene un tavolo, se il robot non è in una posizione contigua al tavolo, se il robot non è scarico.

Per essere eseguita questa azione richiede 10 unità di tempo più 2 unità di tempo per ciascun contenitore di bevanda vuoto che deve essere rimosso più 3 unità di tempo per ogni porzione di cibo consumata.

Se l’azione fallisce per qualcuno dei motivi sopra riportati impiega 30 unità di tempo.

**EmptyFood(x,y)**

Con questa azione il robot svuota gli avanzi di cibo raccolti fino a quel momento nel TrashBasket che si trova nella cella (x,y). L’azione per poter essere eseguita richiede che il robot sia in una delle quattro celle adiacenti (secondo le 4 direzioni) rispetto alla cella (x,y) in cui si trova il TrashBasket. Pertanto l’azione fallisce se la cella (x,y) non contiene un TrashBasket e se il robot non è in una posizione contigua al TrashBasket. L’azione non ha effetti significativi (si veda la parte dedicata alle penalità) se non ci sono residui di cibo da scaricare.

Questa azione richiede 6 unità di tempo per essere eseguita.

**Release(x,y)**

Con questa azione il robot rilascia i contenitori (vuoti) delle bevande che ha a bordo nel RecyclableBasket che si trova nella cella (x,y). L’azione per poter essere eseguita richiede che il robot sia in una delle quattro celle adiacenti (secondo le 4 direzioni) rispetto alla cella (x,y) in cui si trova il tavolo. Pertanto l’azione fallisce se la cella (x,y) non contiene un RecyclableBasket e se il robot non è in una posizione contigua al RecyclableBasket. L’azione non ha effetti significativi (si veda la parte dedicata alle penalità) se non ci sono contenitori vuoti di bevande da scaricare.

Questa azione richiede 8 unità di tempo per essere eseguita.

**CheckFinish(x,y)**

Con questa azione il robot attiva una speciale procedura di controllo (sensing action) per verificare che le consumazioni servite al tavolo in posizione (x,y) siano state consumate. Per poter effettuare l’azione di sensing, il robot deve essere in una delle quattro celle adiacenti (secondo le 4 direzioni) rispetto alla cella (x,y) in cui si trova il tavolo.

L’azione fallisce anche se la cella che viene ispezionata non contiene un tavolo.

Questa azione richiede 40 unità di tempi per essere eseguita.

**Wait**

Questa pseudo azione serve a modellare uno stato di quiete del robot che sta fermo in una cella per 10 unità di tempo.

**Inform (table, request-time, answer)**

Questa azione serve a modellare l’invio da parte dell’agente ad un tavolo table (che aveva fatto un ordinazione specificata tramite request-time) di una risposta sull’accettazione o meno dell’ordinazione.In particolare l’agente robotico può

* mandare un inform di “accepted” . Questo messaggio è adeguato se l’agente stesso sa che il tavolo è già stato pulito (od era inizialmente pulito) ed la richiesta di una ordinazione arriva dopo che il tavolo è pulito.
* mandare un inform di ““delayed”. Questo messaggio è adeguato se l’agente stesso sa che dal tavolo stesso è pervenuta prima una richiesta di pulire in tavolo e poi una richiesta di nuova ordinazione.
* mandare un inform di “rejected” (nella versione attuale questa riposta non è mai adeguata).

Questa azione di Inform richiede una unità di tempo per essere eseguita.

Si noti che fisicamente questa azione non fallisce mai, ma l’emissione di inform non adeguate al contesto sono severamente valutate del meccanismo che assegna penalità

## CONOSCENZA A PRIORI SULL’AMBIENTE

L’agente robotico ha inoltre della conoscenza a priori sull’ambiente in cui si trova ad operare; in particolare conosce:

* la mappa dell’ambiente. Questo implica che il robot conosce la disposizione dell’ambiente, quindi la posizione delle pareti, dei dispenser, dei tavoli, dei seat, di TrashBasket, di RecyclableBasket. Sa quindi anche quali sono le celle libere (quelle iniziali).
* la collocazione dell’area di parking. In particolare al momento iniziale il robot si trova nell’area di parking ed è scarico (non ha alcuna bevanda o cibo a bordo e non ha neppure rifiuti a bordo).
* All’inizio tutti i tavoli sono clean
* All’inizio gli eventuali clienti sono tutti seduti.
* Non è stata fatta nessuna ordinazione

La conoscenza a priori è utile non solo all’istante iniziale, ma anche durante le attività del robot perché vengono fatte alcune assunzioni di persistenza. In particolare non cambiano di posizione le pareti, i dispenser, i tavoli, i seat, i TrashBasket, i RecyclableBasket.

Come risulta evidente da quanto detto sopra l’agente robotico opera in un ambiente complesso (per quanto drasticamente semplificato) in cui vi sono altre entità che possono essere considerate a tutti gli effetti degli agenti deliberativi.

In primo luogo i clienti possono fare ordinazioni per cui possono essere visti come dei generatori di task dal punto di vista del robot. Inoltre i clienti possono (ma non sono tenuti) a segnalare che hanno terminato le loro consumazioni e quindi il tavolo da loro occupato diventa “libero” ma da ripulire.

Inoltre gli operatori umani condividono con il robot lo stesso spazio fisico del caffè in cui si muovono. Si assume che gli agenti umani abbiano un atteggiamento cooperativo con agente robotico, per cui quando sono in movimento e si vengono a trovare in una cella adiacente a quella del robot non vanno intenzionalmente ad urtare il robot, ma stanno fermi fino a quando il robot non si allontana (ma è compito del robot non intralciare i clienti, si veda parte delle penalità) e possono continuare nel cammino da loro previsto.

## I MESSAGGI

Come accennato sopra, la presenza di una molteplicità di attori dotati di autonomia impone che gli attori si scambino dei messaggi.

Per semplificare la situazione, si assume che sia le ordinazioni che il fine consumazione siano inviati dai tavoli (anche se i veri attori sono i clienti degli specifici tavoli ) all’agente robotico.

In particolare il messaggio avrà come emittente un tavolo e conterrà tutte le ordinazioni (sia di food che di drink) di tutti i clienti seduti a quel tavolo (come detto prima, al max 4 clienti per tavolo). Ogni cliente può al max ordinare un food e un drink.

Il tavolo può mandare l’ordinazione solo se il tavolo è pulito oppure, se è sporco, in precedenza era stato mandato un messaggio di fine consumazione.

Pertanto, se il tavolo (i clienti) vuole fare una ordinazione suppletiva, il tavolo deve prima mandare un messaggio di fine consumazione e poi mandare un messaggio di nuova ordinazione.

Si noti che i messaggi sono ricevuti dall’agente solo alla fine dell’esecuzione dell’azione in corso, cosi come le percezioni (si veda sotto).

Per quanto riguarda i messaggi inviati da agente robotico, questi sono modellati con una specifica azione di tipo “inform” descritta in precedenza.

### LE PERCEZIONI

Come nello schema classico dell’architettura ad agente, ad ogni ciclo il robot riceve percezioni dal mondo esterno tramite i suoi sensori. Si suppone che l’agente abbia un sistema di visione omnidirezionale posizionato sul robot in grado di vedere le otto celle contigue rispetto alla cella in cui si trova. In modo più preciso, se il robot si trova nella cella di coordinate (r,c) e la sua direzione è north riesce a vedere cosa contengono le celle (r+1,c-1), (r+1,c) (r+1,c+1) (r,c-1), (r,c), (r,c+1) (r-1,c-1), (r-1,c) (r-1,c+1). Analoghe considerazioni valgono per le altre direzioni.

Il sistema di visione fornisce anche una interpretazione delle celle e si suppone che tale processo di interpretazione sia sempre preciso e affidabile per cui al robot arrivi informazioni corrette sul tipo delle celle circostanti in termini di

* Wall
* Table
* Seat
* FoodDispenser
* DrinkDispenser
* RecyclableBasket
* TrashBasket
* Person
* Empty
* Parking

Si noti che il robot riceve questo tipo di percezioni qualunque sia l’azione che viene eseguita (per semplicità anche una Wait o una Inform).

Le percezioni sono corrette, ma riguardano solo il tipo dell’oggetto, non il suo identificatore: ad esempio il robot è in grado di distinguere un seat da un table, ma non si identificare lo specifico table.

Se il robot tenta di fare una azione di **Forward**in una cella che non è libera riceve la percezione di *bump.*

Le azioni di

* LoadDrink(x,y),
* LoadFood(x,y).
* DeliveryFood(x,y)
* DeliveryDrink(x,y)

danno origine anche a percezioni relative al fatto che il robot abbia un carico o no. Infatti si suppone il robot abbia una sensore di carico, ma questo è poco sensibile per cui l’informazione è solo binaria (scarico o carico)

Il robot ha la possibilità di acquisire deliberatamente informazioni aggiuntive con apposite azioni di sensing. Si veda la descrizione già riportata sopra dell’azione **CheckFinish(x,y).**

# IL TASK E LA VALUTAZIONE DELL’UTILITA’

Come detto nell’introduzione all’agente robotico è assegnato il compito generico di servire ai tavoli di NG Cafe, considerando le seguenti attività:

* Acquisire le ordinazioni da parte dei clienti di un tavolo
* Portare al tavolo quanto ordinato dai clienti di quel tavolo
* Sparecchiare il tavolo, una volta che i clienti hanno finito la consumazione
* Deporre gli avanzi di cibo
* Riporre i contenitori vuoti delle bevande

Anche se il problema ha uno stato iniziale preciso (vedi sotto), potenzialmente non c’è una fine del compito perché l’attività è continua e quindi ci troviamo in una situazione tipica di *on-line continual planning.*

Infatti i messaggi che arrivano dii clienti (nella modellizzazione semplificata dai tavoli) possono essere visti come generatori di un goal di alto livello che il robot deve soddisfare, anche se in molti casi il goal non può essere immediatamente raggiunto perché ci sono altri goal pendenti (ad esempio due messaggi di ordinazione vengono inviati da due tavoli distinti a breve distanza di tempo).

Ovviamente è possibile avere contemporaneamente attivi goal relativi a ordinazioni e goal di andare a ripulire un tavolo o di andare svuotare il robot dei rifiuti raccolti.

Non esiste a priori una gerarchia fissa di goal, ma il comportamento dell’agente robotico deve tenere conto dell’utilità.

Per fornire una qualche indicazione su come impostare le strategie per la scelta degli obiettivi (e alla fine avere un metro comparativo per valutare la bontà del comportamento del robot ) viene fornita una tabella di penalità.

* 50 punti di penalità per ogni istante di tempo trascorso tra una ordinazione di un tavolo e la risposta dell’agente robotico sulla accettazione o meno dell’ordinazione.
* 2 punti di penalità per ogni istante di tempo di attesa e per ogni ordinazione (cibo o bevanda) non ancora consegnata al tavolo se l’ordinazione è stata accettata (accepted). Ovviamente la penalità cessa di essere applicata quando la specifica ordinazione è stata consegnata.
* 1 punto di penalità per ogni istante di tempo di attesa e per ogni ordinazione (cibo o bevanda) non ancora consegnata al tavolo se l’ordinazione è stata accettata ma in modo delayed. Ovviamente la penalità cessa di essere applicata quando la specifica ordinazione è stata consegnata
* 3 punti di penalità per ogni istante di tempo di attesa che il tavolo venga ripulito dopo che è stato dichiarato il finish. Ovviamente la penalità cessa di essere applicata quando l’operazione di CleanTable è terminata.
* 20 punti di penalità per ogni istante di tempo in cui una persona deve stare ferma per lasciare transitare il robot

Mentre almeno in parte queste penalità sono inevitabili nel senso che l’agente robotico non può fare più attività in parallelo, vi sono altre penalità che l’agente può facilmente evitare con un comportamento “smart”.

* 10.000.000 punti di penalità se il robot va ad urtare contro un ostacolo (inclusa una persona).
* 5.000.000 punti di penalità se l’agente rifiuta una ordinazione (quando è nella situazione di accettarla)
* 500.000 punti di penalità se agente risponde che l’ordinazione è delayed (quando è nella situazione di accettarla) e in genere quando manda informazioni sbagliate (esempio manda inform anche se non c’è nessuna richiesta, manda delle inform al tavolo sbagliato)
* 10.000 punti di penalità se manda delle inform inutili (ha già risposto)
* 500.000 punti di penalità per ogni consegna di cibo e/o bevanda al tavolo sbagliato
* 1.000.000 punti di penalità se l’agente svuota gli avanzi di cibo nel RecyclableBasket
* 100.000 punti di penalità se l’agente svuota i contenitori vuoti delle bevande nel trash basket
* 500.000 punti di penalità se il robot tenta di compiere una azione di LoadDrink(x,y), LoadFood(x,y), DeliveryFood(x,y), DeliveryDrink(x,y), CleanTable(x,y), EmptyFood(x,y), Release(x,y), CheckFinish in una locazione non ammessa.
* 100.000 punti di penalità se il robot tenta di compiere una azione di LoadDrink(x,y), LoadFood(x,y) quando l’agente è già a pieno carico.
* 100.000 punti di penalità se il robot tenta di compiere una azione di DeliveryDrink(x,y), DeliveryFood(x,y), quando non c’è food o drink a bordo
* 10.000 punti di penalità per operazioni di CheckFinish inutili (esempio il tavolo è già pulito)
* 100.000 punti di penalità per operazioni di CheckFinish sbagliate (ad esempio CheckFinish fatta prima che sia completato il delivery al tavolo di tutto quanto richiesto nell’ordinazione)
* Se la termine del periodo prefissato ci sono ancora delle ordinazioni a cui non è stato data attenzione, ci sono 500.000 punti di penalità per ogni richiesta di cibo e/o bevanda non eseguita
* Se la termine del periodo prefissato ci sono delle richieste di clean table pendenti, ci sono 500.000 punti di penalità per ciascuna richiesta pendente.

Visto che le penalità dipendono molto dalla dimensione temporale, è ovvio che le prestazioni dell’agente robotico non dipendono solo dalle strategie adottate ma anche (ed in modo molto significativo) dalla sequenza di richieste che arrivano e dalla loro collocazione temporale. Quindi il punteggio non deve essere preso in assoluto, ma relativamente alla dinamica delle richieste e all’ambiente: un agente robotico ha maggiore difficoltà a servire 2n tavoli invece che n; inoltre un ambiente con molti ostacoli è più difficile da navigare che un ambiente delle stesse dimensioni con pochi ostacoli. Infine i movimenti del robot possono essere rallentati se i clienti si muovono molto.

# L’IMPLEMENTAZIONE

Al fine di semplificare lo sviluppo concettuale e implementativo di **NG CAFE**, si mette a disposizione degli studenti un ambiente in CLIPS per lo sviluppo dell’intero sistema (agente + simulatore dell’ambiente). Questo ambiente prevede tre moduli:

* MAIN è usato esclusivamente per la comunicazione tra gli altri due moduli
* ENV che modella l’ambiente in cui opera il robot (compresi i movimenti dei clienti umani)
* AGENT che modella il supervisore del robot

## MAIN

Nel modulo MAIN sono definite le strutture che sono condivise tra il modulo AGENT ed il modulo ENV. Tra le strutture condivise ci sono le percezioni.

In particolare “perc-vision” rappresenta le percezioni visive raccolte dal robot quando il robot esegue una qualunque azione. Come detto sopra, la percezione visiva copre un’area di tre celle per tre celle dove al centro (perc5) c’è la percezione relativa alla cella in cui si trova il robot, in perc1 la percezione relativa alla cella in alto a sinistra mentre in perc9 quella in basso a destra.

Le nozioni di alto e basso, destra e sinistra sono relative alla direzione del robot.

(deftemplate perc-vision

(slot step)

(slot time)

(slot pos-r)

(slot pos-c)

(slot direction)

(slot perc1 (allowed-values Wall Person Empty Parking Table Seat TrashBasket

RecyclableBasket DrinkDispenser FoodDispenser))

(slot perc2 (allowed-values Wall Person Empty Parking Table Seat TrashBasket

RecyclableBasket DrinkDispenser FoodDispenser))

(slot perc3 (allowed-values Wall Person Empty Parking Table Seat TrashBasket

RecyclableBasket DrinkDispenser FoodDispenser))

(slot perc4 (allowed-values Wall Person Empty Parking Table Seat TrashBasket

RecyclableBasket DrinkDispenser FoodDispenser))

(slot perc5 (allowed-values Wall Person Empty Parking Table Seat TrashBasket

RecyclableBasket DrinkDispenser FoodDispenser))

(slot perc6 (allowed-values Wall Person Empty Parking Table Seat TrashBasket

RecyclableBasket DrinkDispenser FoodDispenser))

(slot perc7 (allowed-values Wall Person Empty Parking Table Seat TrashBasket

RecyclableBasket DrinkDispenser FoodDispenser))

(slot perc8 (allowed-values Wall Person Empty Parking Table Seat TrashBasket

RecyclableBasket DrinkDispenser FoodDispenser))

(slot perc9 (allowed-values Wall Person Empty Parking Table Seat TrashBasket

RecyclableBasket DrinkDispenser FoodDispenser))

)

La percezione **perc-load** viene restituita dal modulo ENV solo quando al passo precedente è stata eseguita una azione di

* LoadDrink(x,y),
* LoadFood(x,y).
* DeliveryFood(x,y)
* DeliveryDrink(x,y)

Nello slot load viene restituito quanto percepisce il sensore di peso sul piano di carico del robot: il valore sarà *no* se il robot in quel momento è scarico, mentre sarà *yes* se il sensore sente il peso di qualcosa sul piano di carico del robot stesso. Si noti che la medesima percezione yes viene fornita tanto nel caso che a bordo robot sia presente un solo cibo (o bevanda) quanto nel caso ci siano più bevande (o cibi)

(deftemplate perc-load

(slot step)

(slot time)

(slot load (allowed-values yes no)) )

Nel caso il robot faccia una azione di forward verso una cella occupata riceve una percezione di bump.

(deftemplate perc-bump

(slot step)

(slot time)

(slot pos-r)

(slot pos-c)

(slot direction)

(slot bump (allowed-values no yes)) )

Nel caso l’agente robotico faccia una azione di CheckFinish sul tavolo di coordinate x, y riceve una percezione di conferma o meno

(deftemplate perc-finish

(slot step)

(slot time)

(slot finish (allowed-values no yes)) )

Altre primitive condivise tra il modulo Agent ed il modulo ENV sono le azioni. La struttura per modellare una azione è

(deftemplate exec

(slot step)

(slot action

(allowed-values Forward Turnright Turnleft Wait LoadDrink

LoadFood DeliveryFood DeliveryDrink CleanTable

EmptyFood Release CheckFinish Inform))

(slot param1)

(slot param2)

(slot param3))

Si noti che alcune azioni non hanno alcun bisogno dei parametri (es turnleft): in questo caso i valori dei parametri è NA (not applicable). Il contenuto dei parametri varia a seconda della azione considerata.

Nel caso di una azione di inform:

* Lo slot param1 contiene il receiver (cioè contiene l’identificatore del tavolo a cui è inviata l’informazione sulla accettazione o meno della richiesta)
* Lo slot param2 contiene l’indicazione del tempo (request-time) a cui è stata fatta la richiesta
* Lo slot param3 contiene l’ informazione inviata, cioè

accepted delayed rejected

L’agente robotico riceve anche dei messaggi (asincroni) dal resto dell’ambiente in particolare

(deftemplate msg-to-agent

(slot request-time)

(slot step)

(slot sender)

(slot type (allowed-values order finish))

(slot drink-order)

(slot food-order))

Si noti che:

* Il request-time indica l’istante di tempo in cui la richiesta è stata effettuata
* Il sender è l’identificatore di un tavolo.
* Il type individua due tipi di messaggi che arrivano all’agente robotico. Se lo slot type contiene il valore order, questo indica che è una ordinazione da portare al tavolo indicato nel sender (l’ordinazione è specificata nel seguito), mentre se type è settato a finish questo indica che i clienti hanno terminato le consumazioni e che quindi il tavolo deve essere pulito. Gli slot drink-order e food-order contengono il numero di bevande e di (porzioni di) cibo richieste. Per ciascuno di questi campi il valore mx è 4. Questi campi sono significativi solo nel caso in cui type sia settato a order.

Infine nel modulo MAIN sono definite le strutture che rappresentano le entità presenti nell’ambiente.

Per i tavoli la struttura prevede sia un identificativo (slot table-id) che la posizione in cui è collocata il tavolo (gli slot pos-r e pos-c).

(deftemplate Table (slot table-id) (slot pos-r) (slot pos-c))

Strutture analoghe valgono anche per le altre entità del dominio:

(deftemplate TrashBasket (slot TB-id) (slot pos-r) (slot pos-c))

(deftemplate RecyclableBasket (slot RB-id) (slot pos-r) (slot pos-c))

(deftemplate FoodDispenser (slot FD-id) (slot pos-r) (slot pos-c))

(deftemplate DrinkDispenser (slot DD-id) (slot pos-r) (slot pos-c))

La descrizione di come è fatto l’ambiente all’istante iniziale è fornito da fatti definibili tramite

(deftemplate prior\_cell

(slot pos-r)(slot pos-c)

(slot type (allowed-values Wall Person Empty Parking Table Seat TrashBasket

RecyclableBasket DrinkDispenser FoodDispenser)))

La posizione iniziale dell’agente è definita dal template

(deftemplate initial\_agentposition(slot pos-r) (slot pos-c) (slot direction))

mentre si assume che all’istante iniziale il robot sia scarico.

Si noti che la descrizione effettiva dell’ambiente è presente nel file initmap.txt che contiene le asserzioni relative alle entità dell’ambiente. Poiché i relativi template sono definiti in MAIN, queste descrizioni sono visibili anche nel modulo ENV e AGENT. Analoga situazione avviene per la posizione iniziale dell’agente.

## ENV

Il modulo ENV è molto complesso perché deve modellare l’evoluzione dell’ambiente che non è solo dovuta alle azioni decise da AGENT, ma anche dagli altri agenti (clienti e tavoli che operano come fossero agenti).

L’informazione su quanti clienti sono presenti in NG-cafè e come si muovono è una informazione dinamica (inizialmente si suppone che tutti i clienti siano seduti) è di pertinenza esclusiva pertinenza del modulo ENV e pertanto solo esso ha tale informazione esplicita (l’agente può osservare la presenza di person solo in modo molto parziale attraverso le perc-vision).

Rispetto ad ENV sono eventi esogeni tanto le azioni decise da AGENT (che ENV viene a conoscere visto che i fatti di tipo exec sono visibili in ENV) quanto le azioni decise da altri agenti. Queste ultime sono in qualche modellate supponendo che ENV venga a conoscenza (in certi istanti di tempo) degli eventi che corrispondono alle richieste (sia di ordinazioni che di finish) dei tavoli e alle decisioni dei clienti in termini di movimento.

Per poter modellare l’evoluzione del mondo senza che questo sia immediatamente visibile a AGENT lo stato completo del mondo, il modulo ENV crea una copia della mappa iniziale e delle stato dell’agente che è solo interna a ENV. Queste strutture interne sono poi modificate da ENV per tenere conto dell’evoluzione dello stato. Si noti che ENV deve tenere conto non solo della mappa e dello stato dell’agente, ma anche dello stato delle richieste e dello stato (posizione) dei clienti.

ENV riceve le informazioni relative agli eventi esogeni leggendo il file hystory.txt che contiene per ciascuno degli eventi esogeni previsti anche una marca (temporale) che indica a che step l’evento occorre.

Si assume che il tempo di esecuzione di una azione sia specifico per ogni azione (come indicato quando le azioni sono state descritte) ma in ogni caso il tempo di esecuzione sia (molto) maggiore del tempo di deliberation dell’agente. Per questa ragione il modulo ENV ha anche compito di incrementare il tempo (time) ed il contatore di ciclo (step).

Ad ogni ciclo AGENT è a conoscenza di entrambi visto che status (definito nel MAIN) è visibile anche in AGENT: inoltre le percezioni emesse da ENV e ricevute da AGENT contengono anche esse l’informazione su ciclo e tempo corrente.

## AGENT

Il modulo AGENT è fornito in una versione assolutamente embrionale perché lo scopo del progetto è proprio lo sviluppo di questo modulo. Nella versione fornita esso definisce solo l’interfaccia di comunicazione (fatti di tipo exec) e ad ogni ciclo deve essere l’utente che fornisce l’azione che il robot deve eseguire.

Anche se la conoscenza di base su come è fatto l’ambiente è condivisa (l’agente robotico ha accesso a questa informazione), nel MAIN è definita solo la struttura inziale dell’ambiente Pertanto AGENT non deve operare su questa struttura condivisa cosi come sullo stato iniziale dell’agente.

Per questa ragione nella versione embrionale dell’AGENT sono presenti due regole (beginagent1 e beginagent2) che hanno il compito di “copiare” in strutture interne di AGENT le informazioni relative alla mappa e alla stato inziale dell’agente. Le strutture (è solo una proposta) sono rappresentate da

(deftemplate K-cell

(slot pos-r)

(slot pos-c)

(slot contains (allowed-values Wall Person Empty Parking Table Seat TrashBasket

RecyclableBasket DrinkDispenser FoodDispenser)))

(deftemplate K-agent (slot step)

(slot time) (slot pos-r) (slot pos-c) (slot direction) (slot l-drink)

(slot l-food)

(slot l\_d\_waste)

(slot l\_f\_waste))

Nell’implementazione di AGENT è consigliabile seguire una strategia di sviluppo incrementale che parta dalle funzionalità di base dell’agente stesso. In particolare si suggerisce di partire con

* l’implementazione delle regole che devono gestire l’acquisizione e l’interpretazione delle percezioni
* l’implementazione delle regole che devono gestire l’acquisizione e l’interpretazione dei messaggi e fornire adeguate riposte per mezzo di “inform”
* l’implementazione di una strategia di path planning per pianificare il percorso dell’agente robotico da una cella all’altra.

Si consiglia inoltre di sviluppare una versione base dell’agente che sia in grado di eseguire correttamente e di completare una richiesta di ordine senza porsi subito l’obiettivo di gestire ordinazioni multiple.

Queste (come la capacità dell’agente di evitare di urtare persone) è da sviluppare dopo aver sviluppato le funzionalità di base.

E’ sicuramente utile che si analizzi con una certa cura cosa fa il modulo ENV in modo da comprendere bene il comportamento dell’ENV specie nel caso in cui AGENT non sia ancora in grado di scegliere al meglio le azioni da eseguire (per cui esegue azioni sbagliate e/o inutili con conseguente accumulo di un gran numero di penalità).

Nell’ultima parte del corso alcuni gruppi hanno sviluppato alcune funzionalità di base ed inoltre è stata messa a punto una interfaccia grafica che permette di seguire l’evoluzione del mondo man mano che l’agente decide di fare azioni. Le informazioni su questi embrioni di agenti sono reperibili sul forum degli studenti in MOODLE e possono essere utilizzati da altri gruppi al fine di velocizzare lo sviluppo del progetto.

LA SPERIMENTAZIONE

Come detto sopra, le prestazioni di AGENT dipendono da molti fattori:

* le strategie adottate da AGENT
* la dimensione e la struttura dell’ambiente
* la sequenza e le frequenza di richieste che pervengono all’agente.

Ovviamente le strategie adottate da AGENT giocano un ruolo cruciale, ad esempio non rispondere alle richieste o rispondere in modo sbagliato porta a penalità inaccettabili. Può essere utile implementare un paio di strategie di diversa complessità per poter valutare se si riscontrano variazioni significative.

Per quanto riguarda la dimensione e la struttura dell’ambiente è necessario provare le prestazioni del sistema in alcuni domini di riferimento. Nel seguito vengono riportati graficamente due semplici domini, che non variano per dimensione ma per numero di tavoli, diposizione dei dispenser, ecc. Si raccomanda di provare il sistema anche in domini di maggiori dimensioni (auspicabile che i diversi gruppi arrivino a formulare una serie di ambienti condivisi in modo che il sistema venga valutato sullo stesso insieme di domini. Per iniziare questa opera di condivisione due mondi sono descritti nel seguito.

Il primo dominio di riferimento è riportato in forma tabellare nel seguito (con indicazione delle righe e delle colonne).

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | w | w | w | w | w | w | w | w | w | w | w |
| 9 | w | S11 | S12 |  |  |  |  |  | S34 |  | w |
| 8 | w |  | T1 |  | TB1 | parking | RB1 |  | T3 | S33 | w |
| 7 | w |  | S13 |  | FD1 |  | DD1 |  | S31 |  | w |
| 6 | w |  |  |  |  |  |  |  |  |  | w |
| 5 | w |  |  |  | S42 | T4 |  |  |  |  | w |
| 4 | w | S21 | T2 |  |  | S41 |  | S21 | T2 | S23 | w |
| 3 | w | S22 | S23 | S24 |  |  |  | S22 |  | S24 | w |
| 2 | w |  |  |  |  |  |  |  |  |  | w |
| 1 | w | w | w | w | w | w | w | w | w | w | w |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

La descrizione CLIPS dell’ambiente sopra riportato è contenuto nel file InitMapDom1.txt e per far girare il sistema deve essere copiato nel file InitMap.txt

Per quanto riguarda gli eventi a cui l’agente deve fare fronte, è compito del modulo ENV gestirli. Tuttavia, ENV non li genera in modo randomico, ma riceve queste da una fonte esterna (il file history.txt).

Questo file contiene sia info relative agli spostamenti dei clienti sia alle loro richieste (ordinazioni).

Per descrivere una possibile sequenza di eventi relativa al dominio contenuto in InitMapDom1, è necessario copia nel file history.txt il contenuto del file historyDom1.txt contiene anche la descrizione inziale di quali siano i clienti e dove sia localizzati.

I fatti

(personstatus (step 0) (time 0) (ident C1) (pos-r 7) (pos-c 3) (activity seated) (move NA))

(personstatus (step 0) (time 0) (ident C2) (pos-r 8) (pos-c 10) (activity seated) (move NA))

(personstatus (step 0) (time 0) (ident C3) (pos-r 3) (pos-c 4) (activity seated) (move NA))

indicano che vi sono tre client (C1, C2 e C3) che inizialmente sono seduti in Seat diversi e di cui gli slot pos-r e pos-c indicano le coordinate.

Per quanto riguarda gli spostamenti dei clienti sono previsti dei cammini che i clienti faranno a partire da un certo istante.

(personmove (step 2) (ident C1) (path-id P1))

(personmove (step 9) (ident C1) (path-id P2))

(move-path P1 1 C1 7 2)

(move-path P1 2 C1 6 2)

(move-path P1 3 C1 5 2)

(move-path P1 4 C1 4 2)

(move-path P2 1 C1 5 2)

(move-path P2 2 C1 5 3)

(move-path P2 3 C1 5 4)

(move-path P2 4 C1 4 4)

(move-path P2 5 C1 3 4)

(move-path P2 6 C1 4 5)

Ad esempio, il primo personmove indica ad ENV che il cliente C1 inizia a muoversi (secondo il tragitto specificato da path-id P1) al ciclo 2 (cioè all’istante in cui ENV esamina la seconda mossa dell’agente).

Anche i percorsi seguiti dei clienti sono descritti nel file historyDom1.txt.

Ad esempio (move-path P1 1 C1 7 2) indica che la prima mossa del path P1 consiste nel tentativo di far muovere il cliente C1 dalla sua posizione attuale nella cella di coordinate 7 e 2.

Analogamente nel file history.txt sono prefissati alcuni eventi che corrispondono a ordinazioni da parte dei clienti nonché eventi di fine consumazione (si ricorda che le richieste arrivano dai tavoli) . Ad esempio

(event (step 1) (type request) (source T4) (food 1) (drink 1))

(event (step 5) (type request) (source T3) (food 3) (drink 2))

(event (step 8) (type finish) (source T4))

(event (step 12) (type request) (source T4) (food 0) (drink 2))

Il primo evento indica che al ciclo1 il tavolo T4 emette una ordinazione (type request) di 1 porzione di cibo (food 1) e di una bevanda (drink 1), mentre il terzo indica al ciclo 8 che T4 (in realtà i clienti di T4) ha terminato la prima consumazione (type finish). Infine il quanto evento indica che al ciclo12 T4 fa una nuova ordinazione.

Si noti che le info contenute in history.txt devono essere fornite in modo appropriato tenendo conto sia dell’ambiente in cui si opera che delle possibili tempistiche .

In particolare le descrizioni del movimento dei clienti devono essere contestualizzate all’ambiente: il cliente non può andare in una cella occupata oppure finire in una cella inesistente.

Per ciascuno event, il modulo ENV genera l’opportuno msg-to-agent con la opportuna etichetta temporale che viene calcolata (on the fly) sulla base dell’indicazione di ciclo contenuta nell’evento. Si noti che il modulo ENV prima di mandare il msg-to-agent verifica che l’evento specificato dal progettista sia possibile in quel contesto. Se non lo fosse l’evento viene ignorato.

Nello specificare la sequenza degli eventi, si deve tenere conto del protocollo (non fare nuova richiesta se prima non si è comunicato che la consumazione precedente è terminata) e dei tempi fisici perché agente possa svolgere il suo compito. La history.txt sopra riportata rispetta questi vincoli se pensata relativamente al dominio 2.

**Per semplificare l’implementazione del sistema si suppone che ad ogni ciclo occorra al più un evento**.

Il secondo dominio di riferimento è riportato nel seguito

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | w | w | w | w | w | w | w | w | w | w | w |
| 9 | w | S11 | S12 |  | w | TB1 |  |  | S34 |  | w |
| 8 | w |  | T1 |  | w | RB1 |  | S32 | T3 | S33 | w |
| 7 | w |  | S13 |  | w | DD1 |  | S31 |  |  | w |
| 6 | w |  |  |  | w | parking |  |  |  |  | w |
| 5 | w |  |  | w | w | w | w |  |  |  | w |
| 4 | w | FD1 |  | w |  |  |  |  |  |  | w |
| 3 | w | DD2 |  | w |  |  | S21 |  | S23 |  | w |
| 2 | w | RB2 |  |  |  |  | S22 | T2 | S24 |  | w |
| 1 | w | w | w | w | w | w | w | w | w | w | w |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

La descrizione in CLIPS del dominio è riportato nel file InitMapDom2.txt