

Relazione Sistemi Autonomi

Progettazione del processo deliberativo per agenti che giocano una partita di scopone scientifico.

Introduzione

Il seguente elaborato definirà un sistema basato su agenti autonomi, i quali si coordineranno per giocare una partita di **Scopone scientifico**. Nell'ambito dei sistemi autonomi gli agenti sono entità in grado di compiere azioni autonomamente e prendere decisioni in base alla percezione del contesto in cui operano.

Questo progetto ha una rilevanza nell'ambito dei sistemi autonomi in quanto permette di definire il *processo deliberativo* di un agente, che nel nostro caso porta il giocatore di scopone a scegliere le carte da giocare in base alle condizioni della partita durante il turno di gioco. Le scelte compiute dal giocatore dipendono dal ragionamento guidato dal contesto e sono vincolate dalle regole del gioco.

Vision

Si vuole modellare un sistema in cui 4 agenti seduti intorno ad uno stesso tavolo giocano una partita di Scopone scientifico.

Nel sistema che vorremmo ottenere ogni agente è in grado di fare ragionamenti e quindi di giocare per vincere la partita. Nello specifico non si vuole implementare una vera e propria intelligenza artificiale, bensì definire delle strategie che meglio si adattino al profilo di gioco del giocatore e al contesto.

Opzionalmente, in ogni momento della partita, il sistema desiderato deve permettere la sostituzione di un agente con un giocatore umano. In questo frangente il giocatore umano dovrebbe avere la percezione di trovarsi davanti ad altri giocatori umani.

Goals del progetto

- Formalizzare il sistema descritto.
- Evindenziare il processo deliberativo in base al contesto.
- Analizzare le problematiche e i punti critici legati al caso di studio.
- Definire i possibili profili di gioco dei giocatori.

Requisiti

Il sistema deve essere realizzato nel rispetto dei vincoli definiti dal regolamento di gioco.

Si considera un mazzo di 40 carte nei valori dei semi italiani: spade-coppe-bastoni-denari.

All'inizio della partita ad ogni giocatore sono distribuite 9 carte casuali dal mazzo; le quattro carte restanti sono posizionate sul tavolo (se queste carte sono però tre re, la partita deve ricominciare).

Un giocatore a turno fa una mossa prelevando una carta dalla propria mano in base ai seguenti vincoli di gioco:

1. se il valore della carta scelta corrisponde a quello di una carta sul tavolo, viene effettuata una presa singola.
2. se la somma dei valori di due o più carte sul tavolo corrisponde al valore della carta scelta, viene effettuata una presa multipla.
3. altrimenti si pone la carta scelta sul tavolo.

La presa del giocatore va a finire in nel mazzetto personale.

Alla fine della propria mossa il giocatore passa il turno.

La partita termina quando non ci sono più carte sul tavolo e tutti i giocatori non hanno più carte in mano. Le carte del turno finale sono raccolte dall'ultimo giocatore ad aver eseguito una presa.

Vince la partita il giocatore che totalizza il punteggio più alto. Il punteggio viene definito seguendo questi criteri:

- **Scope:** il giocatore che con una presa raccoglie tutte le carte sul tavolo ottiene un punto.
- **Carte:** il giocatore che ha un mazzetto con almeno 21 carte ottiene un punto. Se vi è un pareggio questo punto non viene assegnato;
- **Denari:** il giocatore che ha nel proprio mazzetto almeno 6 carte di denari ottiene un punto.
- **Settebello:** il giocatore che ha preso il sette di denari ottiene un punto.
- **Primiera:** il giocatore che ha nel proprio mazzetto tutti e quattro i sette ottiene un punto (regola semplificata rispetto allo Scopone canonico).

Analisi dei requisiti

In questo progetto definiamo come **contesto** la base di conoscenza propria dell'agente che gli permetterà di intraprendere le scelte in base all'ambiente e all'evolversi della partita.

L'**ambiente** è il tavolo da gioco su cui sono disposte le carte; rappresenta quindi un insieme di informazioni condivise accessibili a tutti gli agenti. Ogni giocatore è a conoscenza di tutte le carte sul tavolo e percepisce quando una carta viene aggiunta/rimossa.

Gli agenti, in base all'ambiente e al contesto, giocano la partita decidendo quale carta utilizzare in modo da prendere una o più carte dal tavolo per aumentare il proprio punteggio. Ogni agente gioca per se, cercando di vincere la partita, rispettando il regolamento.

Glossario dei termini

Entità	Descrizione
Partita	Insieme di azioni svolte dagli agenti per simulare una partita reale di scopone
Turno	Momento in cui il controllo della partita passa all'agente il quale pensa ed esegue azioni volte a vincere
Player, giocatore, agente	Entità autonome impegnate nel raggiungimento di un obiettivo in base al contesto
Tavolo	Spazio fisico (ambiente) di interazione tra giocatori
Mazzo	Insieme delle carte presenti nel gioco. Sono distribuite tra carte in mano e carte sul tavolo
Carte in mano	Insieme delle carte che possono essere scelte dal player per fare una presa
Carte sul tavolo	Insieme delle carte che potrebbero essere prese dal player in base alle regole e alle carte in mano
Presa singola/multipla	Azione svolta dal giocatore che pone la carta scelta dalla mano e una o più delle carte del tavolo, nel mazzetto personale
Mazzetto personale	Insieme delle carte prese durante la partita
Punteggio/score finale	Somma di tutti i punti accumulati durante la partita. Insieme delle scope e dei punti dipendenti dalle carte presenti nel mazzetto

Qui di seguito analizzeremo i diversi componenti del sistema dal punto di vista di struttura, interazione e comportamento.

Carta

Le 40 carte appartengono al seguente mazzo:

Spade	Coppe	Bastoni	Denari
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	8	8
9	9	9	9
10	10	10	10

Struttura

```
public enum Seed { SPADE, COPPE,  
                  DENARI, BASTONI}  
  
public interface ICard(){  
    public int getNumber();  
    public Seed getSeed();  
    public String getCardStr();  
}
```

Tavolo

Rappresenta uno spazio fisico su cui sono disposte le carte.

Struttura

```
public interface ITable() {  
    /**  
     * Usata per impostare le carte sul tavolo  
     * ad inizio partita.  
     * @param birthCard = carte ad inizio partita  
     */  
    public void setCardsOnTable(List<ICard> birthCard);  
    public List<ICard> cardsOnTable();  
    /**  
     * Esegue la mossa specificata dall'agente,  
     * usando la carta card e prendendo dal tavolo  
     * le carte specificate nella lista taking.  
     * @param card = carta usata nella mossa  
     * @param presa = carte che si desidera raccogliere dal tavolo  
     * @return false se l'azione non è permessa dal regolamento  
     */  
    public boolean action(ICard card, List<ICard> taking);  
    public void printTableCards();  
}
```

Interazione

Ad ogni turno gli agenti interagiscono con il tavolo da gioco, posizionando una carta e se possibile effettuando una presa. Una presa permette di raccogliere una o più carte dal tavolo.

Comportamento

- `cardsOnTable()` ritorna l'insieme delle carte in gioco.
- `addCard(ICard card)` pone una carta sul tavolo e nel caso di una presa ritorna il relativo insieme di carte.

Agente (giocatore)

Per definizione un agente software è un'entità attiva che incapsula un criterio di scelta delle proprie attività (task) al fine di autogestire il proprio flusso di controllo. Lo scopo della sua esecuzione è quello di perseguire un goal, cioè raggiungere un determinato stato.

I quattro agenti da noi considerati operano all'interno di un ambiente condiviso (tavolo), percependo una rappresentazione esplicita ma limitata dei cambiamenti dovuti alle variazioni nel piano delle azioni collettive.

Ogni agente è proattivo, in quanto delibera il proprio corso di azioni basandosi sulla rappresentazione che ha del mondo. Inoltre è un "essere sociale" perché comunica con altri agenti al fine di ottenere/emettere informazioni utilizzando una loro semantica di comunicazione.

Nel nostro caso un agente è quindi un'entità autonoma che prende decisioni in base al profilo assegnato (considerato come "infuso", caratterizzante del giocatore) all'inizio della partita. Ha come goal principale quello di vincere la partita giocando secondo le regole di gioco.

Struttura

```
public interface IPlayerAgent() {  
    /**  
     * @return nome del giocatore  
     */  
    public String getName();  
    /**  
     * Gli sono passate le carte con cui  
     * il giocatore inizia la partita.  
     * Ha effetto solo durante l'init  
     * @param cards  
     */  
    public void setCardsOnHand(List<ICard> cards);  
    /**  
     * Gli è comunicato l'inizio del proprio turno  
     */  
    public void notifyYourTurn();  
    /**  
     * Processo deliberativo che porta l'agente a  
     * prendere una decisione in base al contesto  
     */  
    void deliberate();  
    /**  
     * Gioca la carta scelta  
     * @param card = carta da giocare  
     * @param taking = presa che si intende effettuare,  
     * lista vuota se non si prende nulla dal tavolo.  
     */  
}
```

```
void playCard(ICard card, List<ICard> taking);  
/**  
 * Comunica al successivo giocatore la fine  
 * del proprio turno.  
 */  
void endTurn();  
}
```

Interazione con gli altri agenti

Ad ogni turno l'agente interagisce con gli altri giocatori:

- ricevendo la comunicazione dell'inizio del proprio turno;
- avvisando tutti se ha fatto scopa.
- avvisando l'agente successivo della fine del proprio turno;

Alla fine della partita

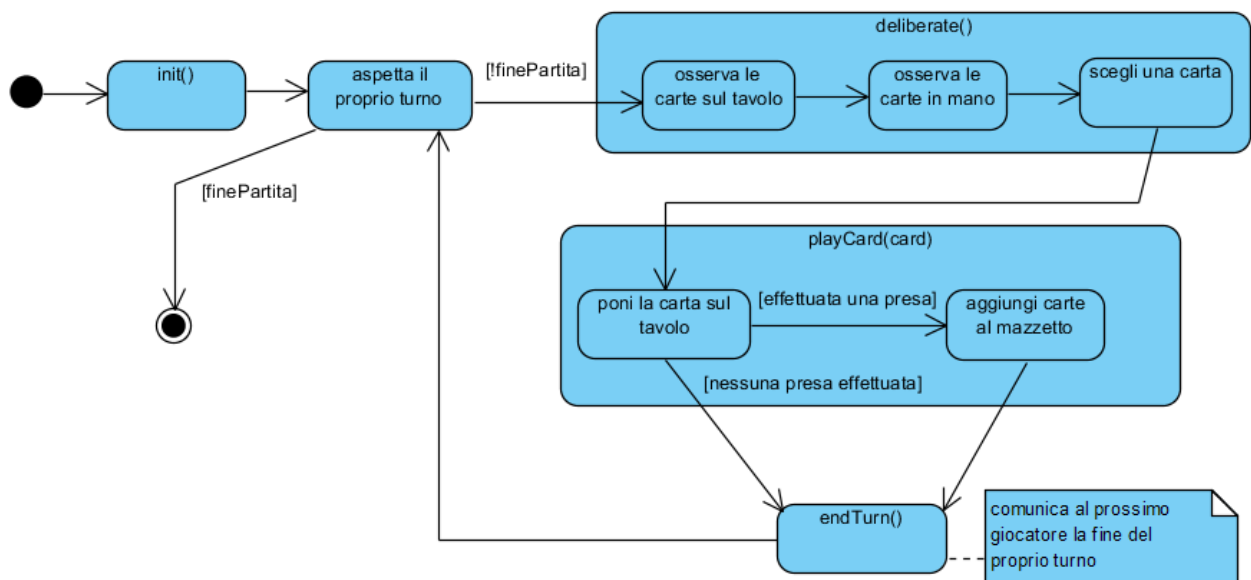
- Tutti gli agenti cooperano per eleggere il vincitore

Interazione con l'ambiente

Ad ogni turno l'agente ha le seguenti interazioni con l'ambiente

- in base alla mossa
 - pone una carta sul tavolo
 - preleva una o più carte dal tavolo (mettendole nel mazzetto personale)

Comportamento



Analisi del problema

Dall'analisi dei requisiti si evince che la complessità dell'elaborato risiede nella modellazione del processo decisionale e cioè nella modellazione del ragionamento dei singoli agenti.

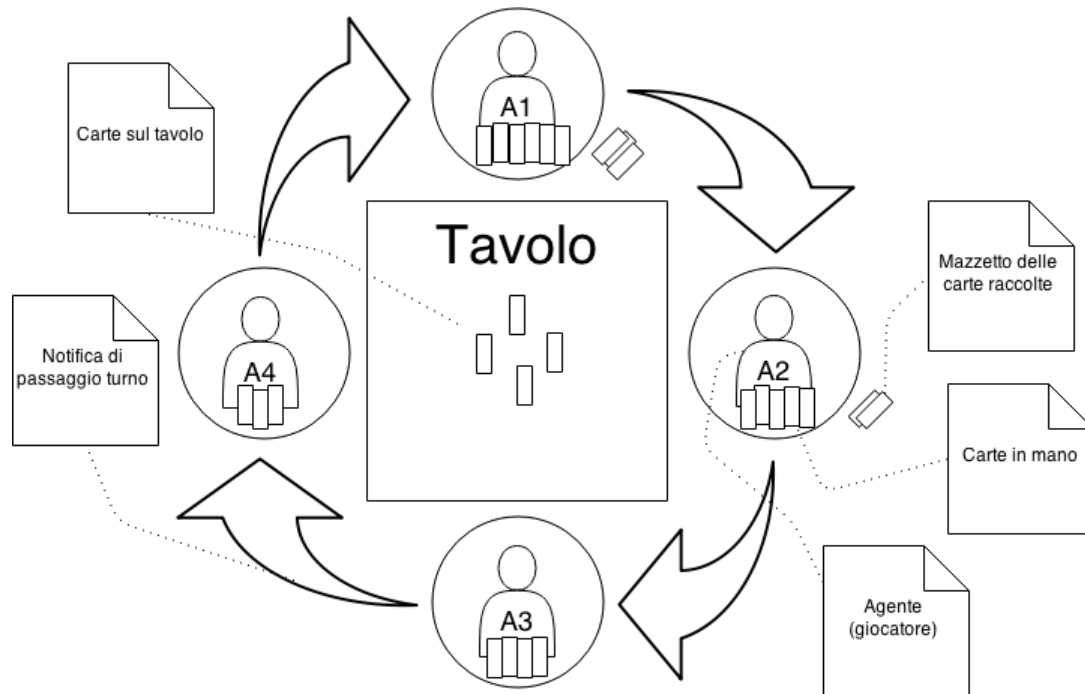
La modellazione del ragionamento dei singoli agenti è fondamentale in quanto porta l'agente a scegliere una determinata carta invece di un'altra anch'essa giocabile, permettendo al giocatore di raggiungere i suoi obiettivi. Il giocatore dovrebbe eseguire scelte non deterministiche, garantendo così comportamenti non prevedibili e variegati, pur rimanendo tali scelte ragionate al fine di aumentare il proprio punteggio.

Per avere un comportamento più verosimile a quello di un giocatore umano, gli agenti non sono vincolati a fare sempre la scelta ottima, bensì possono scegliere in modo casuale tra un set di scelte possibili.

Nello specifico si definiranno dei giocatori che dimostrano un comportamento non prevedibile variabile in base al contesto di gioco (carte che hanno in mano e le carte disponibili sul tavolo), cioè in grado di compiere scelte in modo **non deterministico** da un set di mosse efficaci. Il set di mosse efficaci viene ragionato dall'agente ad ogni turno in base al proprio contesto e all'ambiente di gioco.

All'inizio della partita si assegna ad ogni agente un *profilo di gioco* in modo da caratterizzarne lo stile di gioco. Tale profilo rappresenta l'intesse del giocatore a guadagnare un determinato punto il prima possibile, come per esempio essere interessato a totalizzare più carte di denari per fare il punto denari invece che cercare di collezionare più sette e quindi rincorrere il punto primiera.

Architettura logica



L'architettura logica del sistema è composta da:

- I quattro agenti, (players)
- L'environnement (tavolo)
 - le carte sul tavolo
 - le carte in mano al giocatore
 - il mazzetto personale delle prese del giocatore

All'inizio del gioco si genera il mazzo di 40 carte. Il mazzo viene mischiato in modo randomico e le carte contenute sono distribuite: 4 sul tavolo e 9 ad ogni giocatore.

La gestione del turno avviene tramite passaggio di messaggi. Il giocatore che ha finito il turno notifica al successivo che ha terminato e che ora può iniziare a pensare cosa fare.

Ad ogni turno il giocatore deve decidere, in base alle carte sul tavolo, alle carte in mano e alle possibili prese, quale carta giocare. Le prese popoleranno il mazzetto personale del giocatore.

La partita termina quando l'ultimo giocatore a cui è stato passato il turno è senza carte avendole già giocate tutte nel suo precedente turno.

Lo score finale viene calcolato internamente da ogni agente e comunicato a tutti.

Alla fine del gioco, cooperativamente, i giocatori eleggono il vincitore che ha totalizzato più punti.

Punti critici

Rielaborando quanto detto precedentemente i punti critici risultano essere:

- Definire i quattro giocatori e il loro comportamento come se fossero giocatori umani che interagiscono in un sistema autonomo.
- Dare l'idea ad un giocatore umano di trovarsi di fronte ad automi non prevedibili.
- Definire il processo deliberativo, cioè i ragionamenti e i passi che hanno portato l'agente a compiere una determinata azione.
- Infondere una personalità ai giocatori che caratterizzi le loro scelte.

Progettazione della soluzione

Da quanto espresso dall'analisi del problema si comprendono quali sono i punti critici e le problematiche che si presentano durante l'implementazione, si nota anche come la complessità del sistema sia concentrata nel concetto di agente. Allo scopo di minimizzare la complessità del sistema si adotta un modello orientato alla programmazione di sistemi multiagente. Il modello Belief-Desire-Intention (BDI) è un modello software orientato alla programmazione di agenti intelligenti, che nello specifico offre un meccanismo di separazione di attività attraverso la selezione di piani. Il modello BDI incapsula il concetto di contesto interno di un agente e prevede una netta separazione tra ambiente esterno ed agenti. Alla luce di questi aspetti un'importante scelta implementativa è stata la scelta di *Jason* come framework per sistemi multiagente. Nella fase di progettazione ci accingiamo a definire quello che rappresenta il focus del progetto, ovvero il processo deliberativo del giocatore, la sua base di conoscenza e come interagisce con gli altri per giocare la partita.

Jason

Jason è una piattaforma sviluppata in Java che permette la creazione di sistemi multi-agente e quindi la programmazione del comportamento dei singoli agenti. È un'implementazione di AgentSpeak (linguaggio di programmazione orientato agli agenti) tramite il quale vengono descritti i comportamenti dei vari agenti secondo il modello Belief-Desire-Intention. Brevemente riportate le caratteristiche principali di un agente esprimibili attraverso Jason.

Base di conoscenza

Contiene tutte le informazioni che l'agente ha in termini di Belief, ovvero credenze.

Autonomia rispetto agli stimoli

L'agente percepisce cambiamenti relativi all'ambiente e come conseguenza aggiorna la sua base di conoscenza. Al centro dell'autonomia relativa agli stimoli, provenienti dall'ambiente, vi è lo stato interno e la decisione dell'agente di considerare o meno tali impulsi.

Autonomia cognitiva

L'agente sceglie cosa fare. La sua scelta sarà ponderata ed è vera e propria convinzione, rappresentata tramite sintassi di belief nella sua base di conoscenza e realizzata mediante l'esecuzione di un piano. L'autonomia cognitiva dell'agente lo porta a modificare i propri beliefs.

Scambio di messaggi

A supporto della comunicazione sono a disposizione delle azioni interne che permettono lo scambio di informazioni tra gli agenti. Un agente decide in maniera indipendente se e come processare le informazioni ricevute.

Struttura

Rappresentiamo qui la struttura dell'agente giocatore. Nel paradigma BDI la struttura di un agente è definita in termini di regole, credenze e possibili piani di esecuzione, di seguito ci accingiamo a definire i diversi belief dell'agente, senza concentrarci particolarmente sui piani, che tratteremo invece nel definire il comportamento del giocatore.

Belief (credenze)

A seconda dello stato dell'ambiente il giocatore percepisce e aggiorna le seguenti informazioni:

```
cardsOnHand([]). //Lista di carte che ha in mano, inizialmente  
vuota  
cardsOnTable([]). //Carte presenti sul tavolo (in gioco)  
deck([]). //Il mazzetto  
nextPlayer(Name). //conosce il giocatore del prossimo turno
```

Il giocatore conosce il turno corrente della partita e aggiorna tale conoscenza allo scaturire di determinati eventi. Conosce anche lo stato della partita.

```
current_turn(1).  
~gameEnded.
```

Ogni giocatore conosce quante scope sono state effettuate e da chi.

```
scope_count(p1,0).  
scope_count(p2,0).  
...
```

I giocatori hanno uno storico della azioni compiute durante lo svolgersi della partita.

```
//action(Name,Card,Taking).  
action(p1,card(10,denari),[card(10,denari),card(10,bastoni)]).  
...
```

Interazione

Gli agenti comunicano via messaggi, scambiandosi il turno in modo circolare. Il passaggio del turno avviene dopo che si è eseguita la mossa. L'agente che ha terminato il turno comunica al successivo giocatore che ha passato il turno. Il giocatore successivo riceve il messaggio ed inizia a giocare.

Ogni volta che un agente compie un'azione, la comunica agli altri, in modo che tutti sappiano cosa sta avvenendo nella partita.

In caso in cui il giocatore riesca a far scopa, comunica con un messaggio in broadcast a tutti gli agenti di aver fatto punto.

Tutti i giocatori si rendono conto dell'avvenuta terminazione della partita in quanto ricevono un messaggio che li avvisa che la partita è finita. Questo messaggio è inviato dal giocatore successivo all'ultimo turno.

Alla fine della partita ogni giocatore calcola il proprio punteggio e lo comunica agli altri. Il primo giocatore che conosce il proprio punteggio e quello dei rispettivi avversari elegge il vincitore.

Comportamento

Il comportamento dell'agente rappresenta il focus del progetto, nel paradigma BDI questo si concretizza nell'esecuzione di piani che l'agente esegue in base alle sue convinzioni, siano queste percezioni o altre credenze. Ricordiamo che al termine della mossa l'agente deve presentare un comportamento non prevedibile all'occhio del giocatore umano, inoltre questi deve agire in relazione ad un profilo che gli è infuso alla nascita e ne caratterizza le scelte.

Plans

Basandosi sulle sue convinzioni, l'agente eseguirà dei piani che definiranno il suo comportamento in fase di esecuzione. Il giocatore esegue dei piani per raggiungere dei propri obiettivi.

Nello specifico un agente eseguirà:

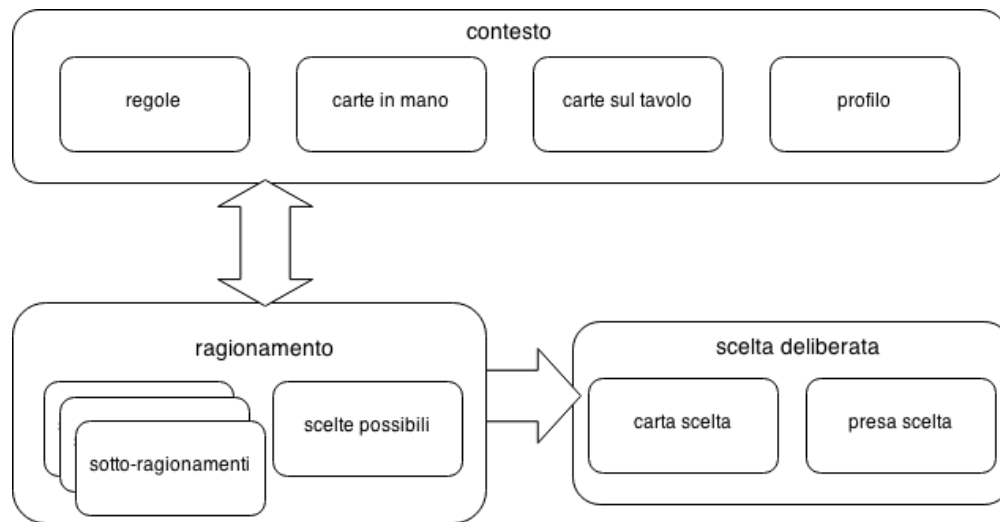
- un piano che lo porterà a valutare le carte in mano e quindi ad eseguire una mossa
- un piano per il calcolo del punteggio finale e per l'elezione di un vincitore

Il fallimento dei piani deve essere gestito; nel caso in cui l'agente non possa compiere un'azione necessaria allo svolgersi della partita, un messaggio di errore deve comunicare lo stato corrente dell'agente. Ogni piano è composto a sua volta da sottopiani che andranno a definire più nel dettaglio le azioni che il giocatore compie prima di portare a termine l'esecuzione di un piano.

```
+!doMove : not gameEnded <- !updateTurn;
                                !execute;
                                !seeIfDone.
+!execute : hasCards <- ?cardsOnHand(Xh);
                                ?cardsOnTable(Xt);
                                !selectAction(Xh,Xt);
                                !checkScopa; //controlla se hai fatto scopa
                                !endTurn.

+!execute : not hasCards <- +gameEnded; !tellEveryoneGameEnded.
```

Processo deliberativo



Dai piani analizzati in precedenza il core del processo deliberativo si concretizza nelle azioni che il giocatore compie per scegliere la carta da giocare (con eventuale presa) in relazione alle carte rimanenti in mano e sul tavolo. L'obiettivo di questo processo non è solo quello di far scegliere al giocatore la mossa più sensata, ma anche quello di fornirgli un grado di errore, così da avere un giocatore che presenti un comportamento non del tutto prevedibile.

Durante il processo deliberativo il giocatore, conoscendo le carte sul tavolo, analizza ogni carta nella sua mano, compiendo una serie di sotto-ragionamenti:

```
+!evaluateCards([Card|Xs],Xt) : true <-
    !evaluateTrustScopa(Card,Xt);
    !evaluateTrustDenari(Card,Xt);
    !evaluateTrustCarte(Card,Xt);
    !evaluateTrustSettebello(Card,Xt);
    !evaluateTrustPrimiera(Card,Xt);
    !evaluateCards(Xs,Xt).
```

Ogni sotto-ragionamento è mirato a calcolare un **coefficiente di fiducia** per la carta selezionata, relativo al particolare sotto-goal del giocatore, come ad esempio può essere il voler massimizzare i denari.

Il coefficiente di fiducia è un valore numerico che rappresenta la fiducia che il giocatore ripone nel giocare quella determinata carta al fine di rincorrere il suo sotto-obiettivo. La sua struttura in termini di belief è la seguente: `trust(card(10,bastoni),denari,0.5)`.

Questi sotto-ragionamenti possono fare uso della base di conoscenza dell'agente e valutare attentamente la fiducia assegnata in relazione a quanto accaduto durante il corso della partita, ad esempio: se più di 21 carte sono state raccolte è inutile provare a massimizzare le carte.

```
+!selectByTrust : true <- ?current_turn(Turn);  
    .findall(trust(Card,Obj,Val), (trust(Turn,Card,Obj,Val)  
& Val > 0.0), L); //tutti i valori di fiducia per il contesto corrente  
    .print("trust list: ", L);  
    !actByTrust(L).
```

Una volta ottenuti i valori di fiducia per ogni carta, l'agente sceglierà in maniera stocastica quale carta giocare, dando maggior priorità alla carta con il coefficiente di fiducia più alto. Le decisioni non sono dunque del tutto casuali ed in parte dipendono dai coefficienti di fiducia ottenuti. In questo senso l'agente ragiona su quale carta porterebbe al raggiungimento del goal (vincere la partita), ma la scelta finale non è comunque strettamente vincolante.

```
+intendedAction(Action) : true <-  
    .print("Azione selezionata: ", Action);  
    playAction(Action); //External action  
    .broadcast(tell,Action). //Notifica gli altri giocatori  
dell'azione compiuta
```

Una volta scelta una carta l'agente interagisce con l'ambiente eseguendo la mossa scelta e compiendo un external action.

Estensione del glossario

Carta giocata	Carta scelta dal giocatore in base al profilo e al coefficiente di fiducia per seguire una strategia
Profilo giocatore	Preferenze di gioco infuse nel comportamento del giocatore per caratterizzarne lo stile di gioco
Coefficiente di fiducia	Valore calcolato in base alle carte presenti sul tavolo e le carte in mano. Rappresenta la fiducia che il player ha nel raggiungere il proprio obiettivo giocando tale carta. (Es. massimizzare denari).

Sotto-ragionamenti

I sotto-ragionamenti servono ad ottenere un valore utilizzato per determinare quanto la carta permette di avvicinarsi al sotto-goal considerato.

Esempio di sotto-ragionamenti:

- Dato il sotto-goal di voler fare denari, il sotto-ragionamento assegnerà un coefficiente di fiducia alto alle carte giocabili che con una presa aumenterebbero il numero di denari nel mazzetto.
- Dato il sotto-goal di voler ottenere il settebello, il sotto-ragionamento assegnerà un coefficiente di fiducia alto alle carte giocabili che con una presa farebbero fare questo punto.
- ...

Profilo giocatore

Il profilo del giocatore rappresenta il carattere del giocatore stesso, questi è importante per poter differenziare il gioco offrendo caratteristiche peculiari ad un determinato agente. Il profilo del giocatore si concretizza nell'importanza che un giocatore dà ad un determinato sotto ragionamento. Sono dei pesi applicabili ai coefficienti di fiducia di ogni singolo sotto-ragionamento e sono esprimibili come:

```
//Esempio di profilo interessato ai denari e primiera  
trust_factor(carte,0.5).  
trust_factor(scopa,0.8).  
trust_factor(settebello,1.0).  
trust_factor(primiera,2.6).  
trust_factor(denari,3.8).
```

Scelte progettuali

Di seguito alcune scelte volte a semplificare la fase di implementazione, non strettamente vincolanti dai requisiti analizzati.

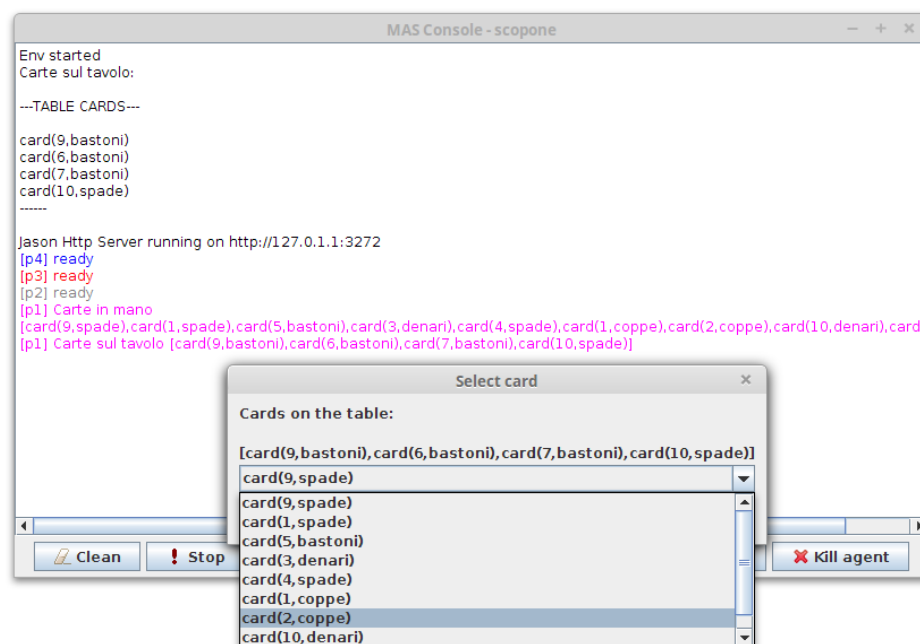
- Ad ogni inizio di partita le carte vengono generate ed ogni agente percepisce quali sono quelle del tavolo e quali sono quelle nella propria mano.
- Il primo giocatore a giocare è il player p1. Il passaggio di turno avviene in modo circolare, p1 passa il turno a p2, p2 a p3, ecc.
- Se sul tavolo non ci sono carte, la scelta della carta da giocare è random.

Implementazione

L'implementazione del prototipo relativo al progetto si suddivide tra classi Java e agenti Jason. Per la realizzazione e il testing è stato utilizzato Eclipse e il plugin Jason for Eclipse.

Nel codice sono presenti 5 agenti autonomi e ognuno ha una caratteristica che lo distingue in base al profilo di gioco che gli si è voluto infondere.

- `basic_player.asl`: È il giocatore che ha uno stile di gioco più equilibrato, non ha un profilo di gioco.
- `card_player.asl`: Gioca cercando di prendere più carte possibili.
- `denari_player.asl`: Cerca di totalizzare il punto denari.
- `primiera_player.asl`: Cerca di totalizzare il punto primiera.
- `human_player.asl`: È un agente che permette l'interazione di un giocatore umano con il sistema autonomo. Per la selezione della carta da giocare è utilizza una GUI, se nessun azione è selezionata, viene scelta una carta a caso.



Il codice del prototipo è reperibile all'indirizzo git:

<https://github.com/pievis/autonomiJason/tree/branch1>

Team di progetto

Federico Torsello, 702619
Pierluigi Montagna, 713248