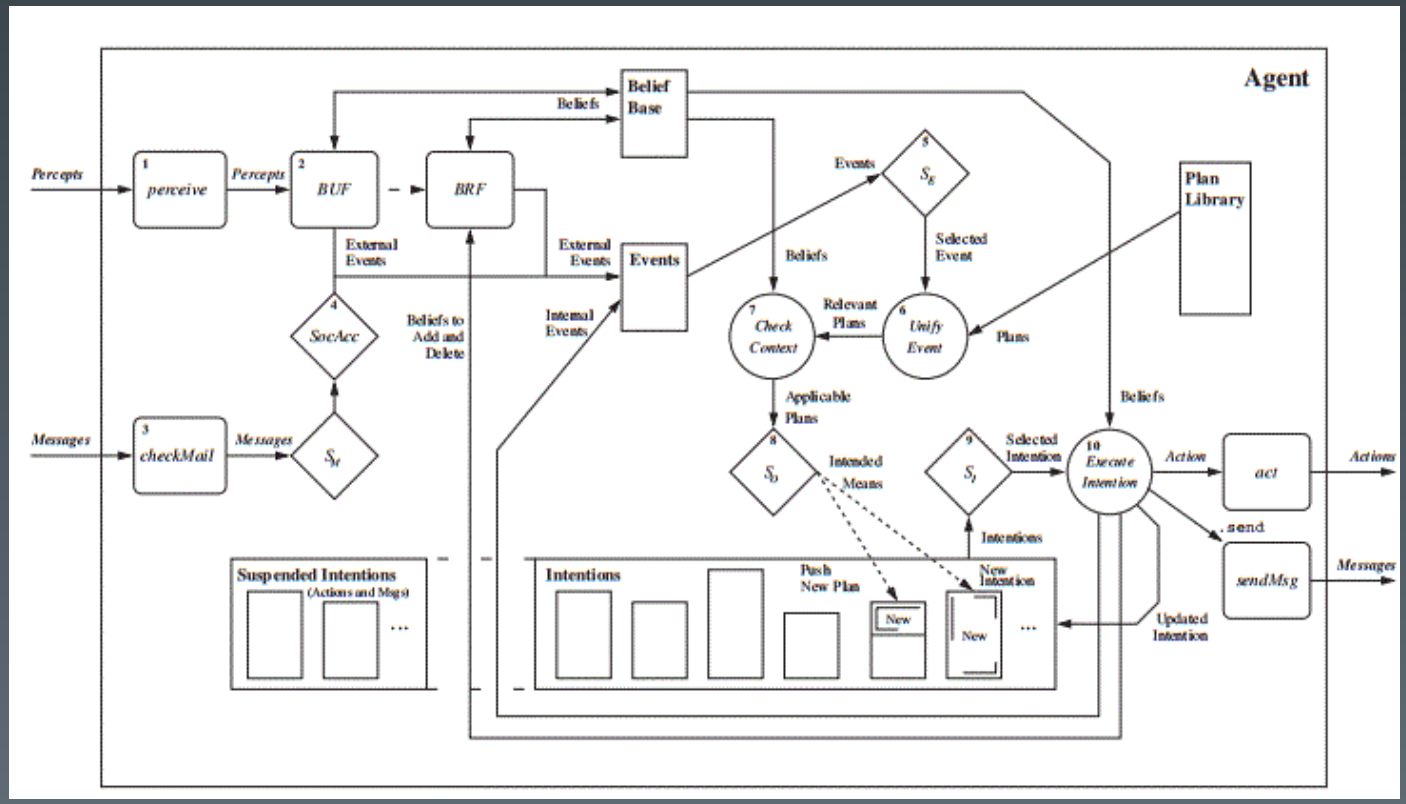
Programming multi-agent system in AgentSpeak using Jason

Capitolo 4 – Jason Interpreter

Il ciclo di Ragionamento

- Un agente opera mediante un ciclo di ragionamento, che nel caso di Jason possiamo suddividere in 10 fasi. L'architettura di un agente Jason è rappresentata in figura:



Fase 1 : Percepire l’ambiente

- La prima cosa che fa un agente,durante un ciclo di ragionamento, è percepire l’ambiente e aggiornare le belief in base all’ambiente.

L’architettura dell’agente deve prevedere una componente, che sia in grado di percepire l’ambiente in una forma simbolica come una lista di letterali.

Il metodo perceive è utilizzato per l’implementare il processo per ottenere tali percezioni.

In un’applicazione, per esempio, in cui l’agente avrà accesso al sensore dati dai dispositivi o sistemi del mondo reale, il metodo perceive dovrà interfacciarsi con loro.

Fase 2 : Aggiornamento delle Belief di Base

- Attraverso una funzione di aggiornamento delle beliefs,che implementa il metodo Buf.

- Il metodo Buf ,di default, presuppone che tutto ciò che è attualmente percepibile dall'ambiente sarà incluso nella lista delle percezioni ottenuta nella fase 1 del ciclo di ragionamento.

- Per aggiornare correttamente le belief,dovremo apportare le seguenti modifiche al metodo Buf:

  - Ogni letterale l in p non è aggiunto in B se è già presente;

  - Ogni letterale l che non è più presente in p viene eliminato anche da B

Con p l'insieme dei letterali ottenuti durante la Fase 1 e B l'insieme delle Beliefs di Base.

Fase 2 : Aggiornamento delle Belief di Base

- Ogni modifica alle Belief di Base attraverso la funzione di aggiornamento delle Belief,porta alla generazione di un evento.

- Un evento è rappresentato come una coppia, dove il primo elemento denota il cambiamento ottenuto mentre il secondo elemento rappresenta l'intenzione associata.

- Se un evento è esterno, non essendo generato da alcuna intenzione, il campo intenzione sarà vuoto e verrà rappresentato da il simbolo T.

- Esempio: Consideriamo un agente che deve percepire il colore di una scatola per la prima volta. L'agente dovrà acquisire delle belief,come ad esempio colour(box1,red) [sorgente(perceive)]. L'evento che potrebbe scatenarsi potrebbe essere il seguente:

  <colour(box1,red)[sorgente(perceive)],T>

Fase 3 : Ricevere comunicazioni da altri agenti

- In questa fase del ciclo di ragionamento, l'interprete verifica la presenza di messaggi che potrebbero essere stati consegnati (da una struttura di distribuzione interna alla piattaforma Jason) all'agente mailBox.

- Il controllo dei messaggi ricevuti avviene attraverso il metodo checkMail(che può essere modificato dal programmatore). Questo metodo ottiene i messaggi ricevuti e memorizzati nell'infrastruttura sottostante e li rende disponibili al livello dell'interprete AgentSpeak.

- Ad ogni ciclo di ragionamento solo un messaggio potrà essere elaborato dall'AgentSpeak.

- La funzione di selezione di un messaggio(SM) selezionerà, tra tutti i messaggi ricevuti e non ancora processati, il primo che sarà processato in questo ciclo di ragionamento.

Fase 4 : Selezionare messaggi “socialmente accettabili”

- Prima che i messaggi vengono elaborati devono passare attraverso un processo di selezione,che determini quali possono essere accettati o meno.
- La funzione che realizza ciò è chiamata funzione di accettazione sociale, ed è realizzata dal metodo SocAcc.
- Di default il metodo SocAcc accetta tutti i messaggi da tutti gli agenti.
- Il ragionamento su quale messaggio accettare o meno dovrebbe essere implementato dall’AgentSpeak, piuttosto che dal metodo SocAcc. Lo scopo del metodo SocAcc è quello di evitare di ricevere comunicazioni da agenti con i quali un agente non dovrebbe avere relazioni.

Fase 5 : Selezionare un evento

- In ogni ciclo di ragionamento, solo un evento in sospeso potrà essere esaminato. Ci potrebbero essere vari eventi in sospeso,causati dal cambiamento di alcuni aspetti dell'ambiente circostante,ma l'agente non ha attraversato abbastanza cicli di ragionamento per poterli gestire tutti.

- La funzione per la gestione di eventi è chiamata funzione di selezione di evento(SE).

- L'insieme degli eventi, in Jason, è implementato come una lista e nuovi eventi vengono aggiunti in coda. Di default la funzione seleziona il primo evento della lista.

Fase 5 : Selezionare un evento

- Esempio: Supponiamo che nell'insieme degli eventi siano presenti due eventi nel seguente ordine
  - < +colour(box,blue) [source(percept)],T >
  - < +colour(sphere,red) [source(percept)],T >

Se la funzione non è modificata, in questo ciclo di ragionamento selezionerà l'evento blue box. Ma se l'agente preferisce gli oggetti di colore rosso rispetto agli altri, la funzione di selezione deve essere riprogrammata cosi da selezionare, in questo ciclo di ragionamento, il secondo evento dalla lista degli eventi.

- Un evento quando viene selezionato è rimosso dall'insieme degli eventi. Se l'insieme degli eventi è vuoto, cioè non ci sono stati cambiamenti nelle belief dell'agente e nei goal dall'ultimo evento manipolato ,allora si passerà alla fase numero nove del ciclo di ragionamento.

Fase 6 : Recupero dei piani rilevanti

- La prima cosa da fare è cercare nella Plan Library tutti i piani rilevanti per un determinato evento.

- Il passo successivo sarà quello di recuperare,dalla Plan Library dell'agente,tutti i piani che hanno un triggering-event che può essere unito all'evento selezionato.

- Una particolare forma di unione(che include le annotazioni) è spiegata attraverso questo esempio:

  Assumiamo di avere come evento selezionato l'evento:
  < +colour(box1,blu) [source(percept)] ,T >

Fase 6 : Recupero dei piani rilevanti

Siano questi i piani presenti nella Plan Library dell'agente:

@p1 +position(Object,Coords) : . . . <- . . . .
@p2 +colour(Object,Colour) : . . . <- . . . .
@p3 +colour(Object,Colour) : . . . <- . . . .
@p4 +colour(Object,red) : . . . <- . . . .
@p5 +colour(Object,Colour)[source(self)] : . . . <- . . . .
@p6 +colour(Object,blue)[source(percept)] : . . . <- . . . .

- Ricordiamo che l'evento selezionato era il seguente:
  < +colour(box1,blue)[source(percept)] ,T >

Fase 6 : Recupero dei piani rilevanti

- I piani rilevanti per l'agente sono { p2,p3 e p6 }

Fase 7 : Determinare i piani applicabili

- Dai piani rilevanti selezionati in precedenza,dobbiamo scegliere quali sono quelli applicabili per l'agente.
- Ovvero conoscendo il know-how dell'agente e le belief attuali, dobbiamo trovare un piano che sembra avere una probabilità di successo.
- Un piano è applicabile se il contesto di ogni singolo piano è una conseguenza logica delle belief di base dell'agente.

Fase 7 : Determinare i piani applicabili

- Esempio: Supponiamo siano questi gli elementi della belief di base:

```
shape(box1,box)[source(percept)].
pos(box1,coord(9,9))[source(percept)].
colour(box1,blue)[source(percept)].
shape(sphere2,sphere)[source(percept)].
pos(sphere2,coord(7,7))[source(bob)].
colour(sphere2,red)[source(percept),source(john)].
```

- Questi sono i piani selezionati in precedenza con i loro contesti:

```
@p2 +colour(Object,Colour) : shape(Object,box) & not pos(Object,coord(0,0))
<- . . . .
@p3 +colour(Object,Colour) : colour(OtherObj,red)[source(S)] & S\==percept &
   shape(OtherObj,Shape) & shape(Object,Shape) <- . . . .

@p6 +colour(Object,blue)[source(percept)] : colour(OtherObj,red)[source(percept)] &
    shape(OtherObj,sphere) <- . . . .
```

Fase 7 : Determinare i piani applicabili

- I piani applicabili per l'agente sono { p2 e p6}

Fase 8 : Selezione di un piano applicabile

- *Il piano applicabile prescelto è chiamato intended, perché è l'insieme delle azioni che l'agente intende(o è impegnato ad) eseguire per gestire l'evento. L'evento è di solito o un obiettivo che l'agente deve raggiungere o un cambiamento dell'ambiente al quale l'agente deve reagire.*

- *Per selezionare il piano applicabile,dall'insieme dei piani applicabili, utilizzeremo una funzione chiamata funzione di selezione del piano applicabile(SO).*

- *La funzione può essere riprogrammata, altrimenti di default sceglie un piano in base all'ordine di visualizzazione nella Plan Library.*

Fase 8 : Selezione di un piano applicabile

- L’ordine dei piani,nella Plan Library,è determinato o da come sono scritti nel codice sorgente dell’agente o dall’ordine con cui i piani sono comunicati all’agente.

- La funzione di selezione di un piano applicabile permette l’aggiornamento dell’insieme delle intenzioni.

- L’aggiornamento può avvenire in modo differente a seconda del tipo di evento.

- Gli eventi esterni potrebbero essere gestiti contemporaneamente e quindi le rispettive intenzioni potrebbero entrare in competizione tra di loro per ottenere l’attenzione dell’agente.

- Gli eventi interni vengono creati quando si aggiungono nuovi goal che l’agente deve raggiungere.

Fase 9 : Selezione di un intenzione per un'ulteriore esecuzione

- *Di solito un agente ha più di un intenzione nel set delle intenzioni, ognuna delle quali rappresenta un punto di attenzione diversa.*
- *Ognuna di queste sono in competizione tra di loro, perché ognuna può essere eseguita in un ciclo di ragionamento, ma solo una per volta può essere eseguita in un ciclo di ragionamento.*
- *Per poter scegliere quale eseguire, utilizzeremo una funzione chiamata funzione di selezione dell'intenzione(SI).*
- *La funzione di selezione dell'intenzione,di default,funziona come il meccanismo di round robin dello scheduling.*

Fase 9 : Selezione di un intenzione per un’ulteriore esecuzione

- Come per gli eventi, l’insieme delle intenzioni è implementato con una lista.

- Quando eseguo un’intenzione, la rimuovo dalla lista dell’insieme dell’intenzione.

- Dopo che ho eseguito l’intenzione, al prossimo ciclo di ragionamento, la inserisco in coda alla lista.

- A meno che il programmatore non intervenga modificando la funzione, di default l’agente dividerà equamente la sua attenzione tra tutte le sue intenzione.

Fase 10 : Esecuzione di uno step di un'intenzione

- *Ci sono tre cose principali che un agente fa ogni ciclo di ragionamento:*
  - *Aggiornare le proprie informazioni sull'ambiente e sugli altri agenti.*
  - *Gestire uno o più eventi generati.*
  - *Agire sull'ambiente(più precisamente agire su una intenzione).*

- *Data un'intenzione particolare è abbastanza semplice decidere cosa fare, dipende dal tipo di formula che appare all'inizio del corpo del piano nella pila dei piani che costituiscono tale intenzione.*

- *Ci sono sei tipi di formule che possono apparire in un corpo di un piano.*

Fase 10 : Esecuzione di uno step di un’intenzione

- Environment Action: Un’azione all’interno del corpo di un piano dice all’agente che dovrà fare qualcosa che avrà effetto sull’ambiente.
- Achievement goal: Indica all’agente nuovi goal da raggiungere, proprio come gli eventi interni.
- Test Goal: In questo contesto ha una particolare funzione: Se ha esito positivo,allora possiamo rimuovere il goal dal corpo del piano e aggiornare le intenzioni all’interno del set delle intenzioni.
- Mental Notes: Se la formula è una belief dovremo aggiungerla o eliminarla dalle belief di base, tutti gli interpreti dovranno passare le proprie richieste al metodo Brf, che apporterà i necessari cambiamenti alle belief di base e genererà i rispettivi eventi. In mancanza di un intervento del programmatore,che crei una differente annotazione, di default alla belief,sia essa aggiunta o rimossa, verrà assegnata come annotazione source(self).

Fase 10 : Esecuzione di uno step di un’intenzione

- Internal action: In questo caso il codice java,fornito dal programmatore, viene eseguito completamente e la formula viene rimossa dal corpo del piano e l’intenzione aggiornata risale nel set delle intenzioni per essere eseguita al prossimo ciclo di ragionamento.
- Expression: Bisogna stare attenti all’uso di espressioni razionali all’interno di un piano piuttosto che in un context. A volte risulta utile usare l’espressione(sopratutto con l’operatore =) per unificare il lato sinistro e destro dell’espressione, ma bisogna stare attenti perché se il valore di ritorno è false allora l’intero piano fallirebbe e le operazioni per il recupero di un piano fallito sono molto costose.

Fase finale prima di riavviare il ciclo di ragionamento

- *Prima che un nuovo ciclo di ragionamento inizi, l'interprete deve effettuare dei controlli, come ad esempio controllare se ha ricevuto dei feedback e se le risposte sono disponibili.*

- *Deve effettuare delle operazioni di pulizia, per esempio delle intenzioni che sono state eseguite completamente.*

- *Se la formula eseguita nel corrente ciclo di ragionamento, è l'ultima presente nel corpo del piano allora il piano è stato eseguito con successo e può essere rimosso dalla cima della pila delle intenzioni.*

Fallimento di un piano

- *Dovremo dotare gli agenti di un piano di emergenza nel caso in cui il piano fallisca.*

- *Indicheremo i piani per il raggiungimento del goal g con la notazione +!g e il piano di emergenza con la notazione -!g ,che ci indica che il piano +!g è fallito.*

- *Ci sono tre cause principali per il fallimento di un piano:*
  - *Mancanza di piani rilevanti o applicabili per il raggiungimento del goal.*
  - *Fallimento di un test goal*
  - *Fallimento di un azione*

Modalità di configurazione ed esecuzione di un interprete.

- *Questo codice rappresenta un esempio della definizione di un sistema multi-agente in Jason:*

```
MAS <mas_name> {
    infrastructure: <Centralised|Saci|Jade|...>
    environment: <environment_simulation_class> at <host>
    executionControl: <execution_control_class> at <host>
    agents: <ag_type1_name> <source_file> <options>
    agentArchClass <arch_class>
    agentClass <ag_class>
    beliefBaseClass <bb_class>
    #<num_instances> at <host>;
    <ag_type2_name> ...;
}
```

- *Nel campo <options> possiamo configurare l'interprete.*

Modalità di configurazione ed esecuzione di un interprete.

- *E' possibile settare il campo option di <options> con uno dei seguenti valori:*

  - *Events: Le opzioni sono* **discard**, *requeue o retrieve.*
  - *IntBels: Le opzioni sono* **sameFocus**, *newFocus.*
  - *Nrcbp: Number of reasoning cycles before perception.*
  - *Verbose: E' un numero tra 0 -* **1-** *2.*
  - *User settings: Gli utenti possono creare le proprie configurazione nella dichiarazione dell'agente,ad esempio:*
    ...agents:ag1[verbose=2,file="an.xml",value=45];

  Posso accedere ai parametri della classe Settings nel seguente modo:

  getSettings().getUserParameter("file");

Modalità di configurazione ed esecuzione di un interprete.

- *Di seguito saranno elencate le diverse modalità di esecuzione disponibili in Jason:*
  - **Asynchronous**: *Tutti gli agenti sono eseguiti in modo asincrono. Un agente passa al prossimo ciclo di ragionamento solo se ha finito quello corrente*
  - Synchronous: *Tutti gli agenti eseguono un ciclo di ragionamento ad ogni passo di esecuzione globale. Quando un agente termina il proprio ciclo di ragionamento informa un controller di Jason ed aspetta un segnale "carry on". Il controller attende che tutti gli agenti abbiano finito il loro ciclo di ragionamento prima di inviare il segnale di "carry on".*
  - *Debugging: Modalità simile a quella sincrona, con l'unica differenza che il controller aspetta la pressione del tasto "step" da parte dell'utente per inviare il segnale di "carry on" agli agenti.*

Modalità di configurazione ed esecuzione di un interprete.

- *E' possibile configurare un debug nella classe .mas2j come in questo esempio:*

```
MAS test {
   infrastructure: Centralised
   environment: testEnv executionControl:
   jason.control.ExecutionControlGUI
   agents: ...
}
```

- *La classe jason.control.ExcutionControlGui implementa il controller Jason, che crea una Gui per il debugging.*

- *Lo strumento grafico che aiuta il debug è chiamato "Jason's Mind Inspector", che consente agli utenti di verificare i cambiamenti mentali di un agente dopo un ciclo di ragionamento.*

Pre-Defined Plan Annotation

- *Jason fornisce delle annotazioni predefinite,che quando sono immesse nelle annotazioni di un piano ne influenzano l'interpretazione.*

- *Le annotazioni predefinite sono le seguenti:*
  - *Atomic: Se un'istanza di un piano con un'annotazione atomica è scelta dalla funzione di selezione di un'intenzione, quest'intenzione potrebbe essere selezionata per essere eseguita nei prossimi cicli di ragionamento finché il piano non è finito.*

  - *Breakpoint: E' utile nel debugging;Se la modalità di debugging è in esecuzione ed un agente inizia l'esecuzione di un'intenzione con un'istanza di un piano,che ha un'annotazione di breakpoint,allora l'esecuzione è interrotta ed il controllo torna all'utente.*

Pre-Defined Plan Annotation

- *All_unifs: E' usata per includere tutte le possibili unificazioni di un piano applicabile nell'insieme dei piani applicabili. Normalmente, per un dato piano,solo la prima unificazione trovata è inclusa nell'insieme dei piani applicabili.*

- *Priority: E' un termine(of arity 1) che può essere usata di default dalla funzione di selezione di un piano, per dare una semplice priorità di ordine ai piani applicabili e alle intenzioni. Più alto è il numero dato come parametro, maggiore sarà la priorità del piano nel caso di piani multipli applicabili.*

The End.