

# **Multi Agent System** with **Natural Language Processing**

Università degli Studi di Bologna

II Facoltà di Ingegneria

Approfondimento Sistemi Multi Agente LS – A.A. 2008/2009

**Danilo Martino**

# INDICE

1. [Sistema TALISMAN come primo modello non sequenziale](#)
2. [Approccio lessico-strutturale vs Approccio linguistico-cognitivo](#)
3. [HERALD, approccio robusto al NLP](#)
4. [KQML, un esempio di linguaggio coordinativo fra Agenti](#)
5. [Sviluppi recenti](#)
6. [Cos'è un CORPUS?](#)
7. [Importanza del linguaggio in sistemi di NLP con TAL](#)
8. [Cos'è un'ONTOLOGIA?](#)
9. [Bibliografia consultata](#)
10. [Bibliografia citata](#)

# SISTEMA TALISMAN, PRIMO MODELLO NON SEQUENZIALE

(1995)

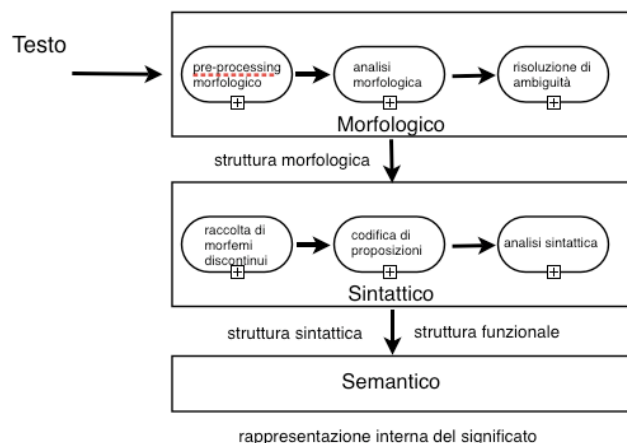
[2][7]

In questo sistema è usato un MAS(Multi-Agent System) che con i suoi paradigmi enfatizza l'utilizzo di moduli e la loro interazione. Tali moduli vengono chiamati agenti. L'intelligenza è distribuita fra tutti gli agenti, entità autonome in grado di agire razionalmente percependo l'ambiente e lo stato della loro conoscenza.

In questa applicazione gli agenti corrispondono ai classici livelli in linguistica (morfologico, sintattico, semantico) o a complessi fenomeni linguistici quali coordinazione, ellissi, negazione ed altri.

Contribuendo alla ricerca di soluzioni sintatticamente corrette, gli agenti facilitano la risoluzione di alcune ambiguità che sono insite nel linguaggio.

Molte applicazioni di NLP usano un'architettura sequenziale basata sui classici livelli linguistici del tipo morfologico, sintattico, semantico, pragmatico, pre-trattamento. Vediamo i limiti di applicazioni che si basano su una struttura sequenziale prendendo in considerazione un progetto chiamato CRISTAL realizzato all'interno di un progetto Europeo chiamato MMI2<sup>1</sup>. Tale sistema è realizzato su tre livelli come mostra la figura sotto:



il primo livello è quello relativo all'analisi morfologica. Le stringhe di caratteri, in ingresso al sistema, vengono convertite in una o più coppie di sequenze (voci lessicali, categorie grammaticali + valori di variabile) che corrispondono alle frasi.

<sup>1</sup> Multi-mode Interface for Man-Machine Interaction, il cui scopo fu quello di realizzare un'interfaccia utente per un sistema informatico esperto. L'interfaccia include modi differenti di espressione: Natural Language, modalità grafica, modalità gestuale, linguaggio comune.

Il secondo livello è quello della struttura grammaticale della frase. L'analizzatore costruisce strutture di frase a partire dalle sequenze prodotte nel livello precedente.

L'ultimo livello è quello della espressione logica: la struttura costituente di una frase è convertita in una struttura operatore-operando.

Il **modulo morfologico** è formato da tre step:

1. fase di *pre-processamento*: i caratteri sono standardizzati ed il testo è diviso in blocchi.
2. tali blocchi sono quindi processati individualmente da un *analizzatore morfologico* che attribuisce una o più interpretazioni ad ognuno dei termini della coppia (voce lessicale, categoria grammaticale).
3. *risoluzione di ambiguità (modello di Markov)*: in base al contesto viene ridotto il numero di interpretazioni multiple rimuovendo le ambiguità.

Il **modulo sintattico** è formato da tre step:

1. *codifica di proposizioni*: il primo stadio taglia le frasi complesse dividendole in clausole mentre, allo stesso tempo, viene costruita la struttura. Le clausole sono unità sintattiche complete e sono più facili da analizzare delle frasi intere.
2. lo scopo dell'*analisi sintattica* è di costruire la struttura sintattica di una clausola.

Il **modulo semantico** costruisce una struttura logica.

Un MAS permette, invece, di avere una completa distribuzione di conoscenza e di risultati parziali attraverso una comunicazione diretta fra agenti (è utilizzato un controllo locale e non globale dell'intero sistema). Ogni agente ha un compito specifico ed una rappresentazione della conoscenza non necessariamente coerente con quella degli altri agenti. Inoltre le capacità di ogni agente di ragionare e di prendere decisioni lo rendono capace di agire autonomamente.

### **Agente nel modello TALISMAN**

Un agente nel sistema TALISMAN è un'istanza semplificata di un agente generico (definito da Demazeau, autore di [2]) che può essere suddiviso in due parti principali: la sua *rappresentazione di conoscenza* e il suo *processo di conoscenza*.

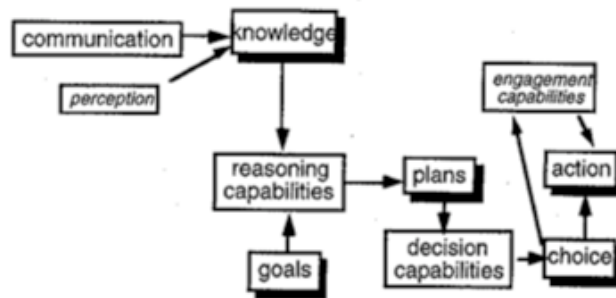
La *rappresentazione di conoscenza* può essere collegata a tipi differenti di conoscenza; espressioni della rappresentazione di conoscenza sono: una rappresentazione del mondo esplicita e astratta in cui gli agenti evolvono; le abilità per rappresentare cosa potrebbe offrire un agente agli altri agenti; i suoi interessi e le abilità degli altri agenti; rappresentazioni astratte ed esplicite di problemi o obiettivi che l'agente deve risolvere; scelte disponibili e decisioni possibili che devono essere prese. Questa conoscenza può essere acquisita o data attraverso la comunicazione con altri agenti.

Il *processo di conoscenza* usa la rappresentazione di conoscenza per produrre le attività di ragionamento dell'agente. Il processo di conoscenza contiene capacità di ragionamento, inclusa l'individuazione di incoerenze, l'uso di dati, il ragionamento sulla conoscenza e sul comportamento di altri agenti, e le capacità di decisione.

E' anche possibile astrarre i goals dall'osservazione del comportamento degli altri agenti.

Questi goals non esistono esplicitamente dentro un agente ma possono essere codificati in algoritmi, dati esplicitamente o acquisiti attraverso la comunicazione con altri agenti.

Dalla conoscenza e dai goals, un agente può essere obbligato a considerare un set di possibili soluzioni o piani per raggiungere l'obiettivo. In realtà possono esistere molteplici soluzioni al problema e fra queste solo una sarà quella definitiva. La scelta, da parte dell'agente, di questa soluzione potrebbe anche essere errata da un punto di vista esterno. Per limitare tali errori potremmo introdurre all'interno del sistema un modulo di capacità di decisione da parte dell'agente. Uno schema del processo di acquisizione di conoscenza e goals, di capacità di ragionamento da cui derivano i piani e le strategie, eventuale capacità di decisione che permette di scegliere la strategia giusta e quindi arrivare all'azione è il seguente:



### Modello TALISMAN

Il modello di TALISMAN prevede l'uso del concetto di **legge** definito da N. H. Minsky. Questa nozione permette l'imposizione di protocolli su sistemi aperti, distribuiti, eterogenei, in sviluppo. Una nuova architettura chiamata **Law Governed Architecture for Distributed System (LGAD)** permette l'esecuzione di protocolli in cui tutte le interazioni fra agenti sono governate da un set esplicito di regole chiamate le **leggi del sistema**. Tali leggi sono *globali* nel senso che tutti gli agenti sono costretti a seguirle e rispettarle, ma possono avere anche valenze assolutamente *locali* ad ogni agente. Le leggi possono essere facilmente aggiornate modificando le regole che le descrivono. La flessibilità del modello LGAD è una conseguenza del fatto che le leggi siano così esplicite permettendo a chiunque di implementare uno specifico comportamento per un'applicazione specifica. Per ogni agente vengono introdotte e definite leggi linguistiche locali, private, e che corrispondono al controllo locale dell'agente. A livello di società le leggi globali regolano la comunicazione. Per ogni messaggio inviato esiste una legge che fornisce una regola per quell'azione. Questa regola può far sì che il messaggio venga consegnato al destinatario originale, che venga bloccato, modificato, fatto tornare indietro o inviato ad una nuova destinazione. Nel sistema TALISMAN esistono diverse leggi, come quelle iniziali e di coordinazione (di blocco, reindirizzamento, trasformazione, priorità, etc.), introdotte per regolare vari conflitti.

### Sistema TALISMAN

La rappresentazione di conoscenza definisce le norme della società e cosa è permesso all'interno di essa, mentre il processo di conoscenza definisce come usare la rete secondo le regole della società. Allo stato attuale dell'arte, nel progetto TALISMAN sono previsti i seguenti agenti: PRET agente per il pre-processamento, MORPH per l'analisi morfologica, SEGM per la suddivisione in clausole, SYNT per l'analisi sintattica, TRANSF per le trasformazioni di enunciati(interrogativi,

imperativi,...) in clausole dichiarative, STAT per la risoluzione di ambiguità statistiche, COORD per la coordinazione, NEGA per le negazioni, ELLIP per ellissi.

Come esempio di agente all'interno del sistema TALISMAN prendiamo in considerazione l'agente SYNT. La rappresentazione di conoscenza di SYNT contiene un dizionario dei verbi, grammatiche, indicatori di strutture lessicali e di strutture grammaticali, preposizioni che introducono frasi avverbiali.

Gli indicatori di strutture grammaticali sono parole appartenenti alle categorie grammaticali la cui funzione è di indicare precisamente l'inizio di una frase:

- articoli determinativi (D) ma in qualche caso nomi o aggettivi (F) che permettono di definire l'inizio di una frase nominale (SN)
- preposizioni (P) che introducono frasi preposizionali (SP)
- pronomi pre-verbali (Y) e verbi (V) che iniziano una frase verbale (SV)

Ad esempio una sequenza del tipo (D/Y F/V) può predire, nell'agente SYNT, un SN o un SV.

Lo scopo dell'agente SYNT è quello di costruire strutture sintattiche di frasi.

Per far ciò quest'agente ha accesso a dati memorizzati in un lessico di espressioni definite come frasi avverbiali e parole composte ed usa una sistema esperto che include l'algoritmo di Earley.

SYNT interagisce con MORPH, SEGM, COORD, TRANSF comunicando a loro le scelte fatte o proponendo soluzioni per la disambiguazione di frasi. La sua memoria interna memorizza informazioni sugli agenti con cui può comunicare e sulle decisioni prese.

## **Protocolli d'interazione**

Prima di parlare protocolli di interazione in TALISMAN ci occuperemo dei protocolli d'interazione in generale.

Ogni agente deve essere in grado di estrarre da ogni messaggio ricevuto porzioni di informazione utile al controllo di interazione con altri Agenti ma in generale con l'intera società.

Un agente non solo dev'essere abile nel recuperare, dal messaggio stesso, le intenzioni del mittente ma deve anche capire come reagire e cosa aspettarsi dopo l'invio del messaggio. Tutti questi requisiti devono essere racchiusi in un unico framework per strutturare le interazioni fra agenti: i protocolli d'interazione.

Sono stati descritti una serie di classici protocolli nella letteratura DAI come il protocollo Contract Net<sup>2</sup>. Più di recente sono stati sviluppati protocolli specifici come Introduction protocol<sup>3</sup> o lo SNAP negotiation protocol<sup>4</sup>. Quest'ultimo protocollo deriva da studi fatti su situazioni di lotta per la sopravvivenza delle società umane.

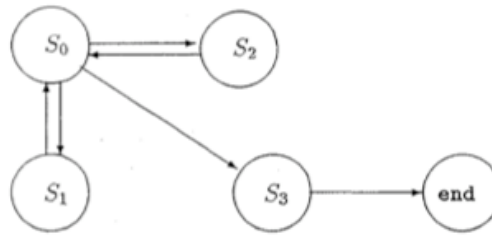
Un esempio di rappresentazione generale di un protocollo d'interazione è quella in figura sotto:

---

<sup>2</sup> R.Davis and R.G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20:63-109, 1983

<sup>3</sup> Sabine Berthet, Yves Demazeau, and Olivier Boissier. Knowing each other better. In 11th International work- shop on Distributed Artificial Intelligence, IWDAI '92, Glen Arbor, Michigan, February 1992.

<sup>4</sup> M. Chang and C. Woo. SANP: A communication level protocol for negotiations. In Werner and Yves Demazeau. editors. Decentralized AI 3. Amsterdam, Netherlands, 1992. 'Elsevier Science Publishers B.V.



E' una rete di transizioni con nodi(stati in cui si può trovare un agente) e rami(interazioni fra agenti). Un agente transita da uno stato all'altro in dipendenza dal tipo di messaggio che riceve. Questo influenza il suo comportamento e in particolare il tipo di messaggio che questo può inviare ed i parametri in esso contenuti.

In realtà il passaggio da uno stato all'altro avviene anche in base a determinate condizioni proprie dell'agente.

Ad esempio, ipotizziamo di avere due agenti che comunicano, A e B, e che quanto scritto è il formalismo per cui può avvenire o meno un passaggio di stato:

*if state s1 & condition c then state s2*

vuol dire che se l'agente A si trova nello stato s1 e la condizione c è verificata, allora l'agente B andrà nello stato s2 dopo aver ricevuto il messaggio da A.

### **Protocollo di conoscenza cooperativa (Sian<sup>5</sup>)**

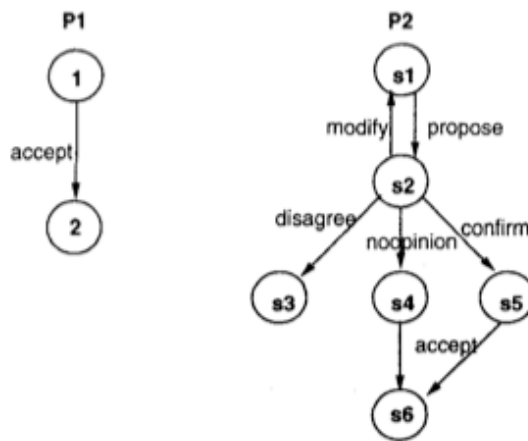
Sian presenta un approccio, per la creazione di un protocollo di comunicazione, per l'acquisizione di informazioni utili nei MAS.

Nel sistema TALISMAN è stato implementato il protocollo di Sian utilizzando un controllore centralizzato ed una struttura di conoscenza comune(lavagna) attraverso cui definire una comunicazione indiretta.

Probabilmente è proprio questa la causa che determina l'esistenza di agenti non completamente autonomi.

Per definire in maniera esplicita lo stato raggiunto dagli agenti durante la conversazione, è stato previsto un accorpamento al protocollo di comunicazione di Sian nei seguenti protocolli di interazione di figura sotto:

<sup>5</sup> Sati Singh Sian. Adaptation based on cooperative learning in multi-agent systems. In Yves Demazeau and Jean-Pierre Muller, editors, Decentralized AI 2. Elsevier Science Publishers B.V., 1991.



Il protocollo P1 è semplice: un agente forza un altro agente per accettare la sua ipotesi.

Il protocollo P2 inizia con un possibile ciclo di proposta-modifica sulle ipotesi in questione dove un agente A invia una richiesta ad un agente B, a cui quest'ultimo risponde. La risposta può essere di modifica (e quindi si ritorna nel ciclo di interazione iniziale) oppure di disaccordo, senza opinione o di conferma e porta l'agente B nello stato s3, s4, s5. Lo stato s3 è uno stato finale della comunicazione. Gli stati s4 e s5 sono stati di accettazione, dove però non è prevista risposta al messaggio inviato.

### Un esempio di interpretazione di frase

Mostriamo l'interazione in TALISMAN con un esempio di cooperazione fra gli agenti PRET, MORPH, SEGM, SYNT, sulla seguente frase:

*Pilots like flying planes can be dangerous*

L'agente PRET esegue una fase di pre-processamento sui caratteri e sul testo, tagliandolo in moduli. L'agente MORPH ha una conoscenza basata su dizionari e grammatiche. Le sue capacità di ragionamento corrispondono a risolvere ambiguità morfologiche. Lo scopo della procedura è trovare una struttura morfologica. I moduli del testo sono quindi processati individualmente da un analizzatore morfologico che attribuisce una o più interpretazioni ad ogni termine di una coppia (entità lessicale, categoria grammaticale) come [pilot/F(noun), V(verb)].

L'agente SEGM suddivide le frasi in clausole trovando anche le loro strutture. Ogni clausola è una unità sintattica completa più facile da analizzare rispetto all'intera frase.

L'agente SYNT è stato già studiato in precedenza.

Vediamo quali sono le interazioni possibili in un'interpretazione di questo tipo.

- protocollo di Sian per la proposta e la conferma di ipotesi o per accettare informazioni.

- il concetto di leggi (Minsky<sup>6</sup>). Le leggi sono globali nel senso che tutti gli agenti devono obbedire, oltre che locali ad ogni agente. Le leggi possono facilmente essere aggiornate attraverso le modifiche di regole che le descrivono. Nel modello in questione esistono differenti tipi di leggi fra cui una iniziale e quelle di coordinazione per risolvere i conflitti (tali leggi permettono di bloccare, reindirizzare, trasformare, assegnare priorità ad un messaggio).

In ogni caso per la risoluzione della frase in esempio, il primo agente ad iniziare l'esecuzione è l'agente PRET che informa tutti gli altri agenti, utilizzando messaggi asincroni, che la frase inizia la fase di pre-trattamento e che si tratta di una frase di tipo assertivo. Questo messaggio consente a tutti gli agenti di accedere allo stadio 2 del protocollo di primo tipo (figura precedente). Quando il messaggio inizia con I è un messaggio informativo, se inizia con H e di ipotesi.

Qualora il messaggio sia di tipo informativo, l'agente MORPH è il primo ad iniziare la sua computazione (in realtà se la frase è interrogativa è l'agente TRANSF a trasformarla secondo un ordine canonico).

L'agente MORPH analizza la frase da sinistra verso destra scoprendo i termini ambigui in modo da attivare le regole morfologiche di disambiguazione. A questo punto l'agente MORPH fa delle ipotesi di risoluzione (esempio *Pilots è un nome*) e le comunica all'agente SYNT che le valuta:

H1 : <MORPH><SYNT><SEGM><asynchrone><request> <propose><P2 (S2) ><ph( [pilots=F] ) >  
 <SEGM> <MORPH><SYNT> <asynchrone><answer><noopinion><P2(S4) ><ph( [pilots=F] ) >  
 <SYNT><MORPH><asynchrone><answer><confirm> <P2(S5)><ph([pilots=F])>  
 <MORPH><SYNT><asynchrone><inform><accept> <P2(S6)><ph([pilots=F])>

Quando nuova conoscenza è validata da un agente, questo modifica la sua memoria locale e riavvia il processo di risoluzione.

In parallelo l'agente SEGM informa l'agente SYNT e MORPH del numero di proposizioni e cioè di verbi nella frase (per ipotesi la frase non ha virgole).

Quest'informazione può aiutare a scegliere il verbo appropriato.

I3. <SEGM> <MORPH, SYNT> <asynchrone> <inform><accept> <P1(2) ><ph(nb-verb=l)>  
 H3. <MORPH><SYNT><asynchrone><request><propose><P2(S1)><ph([like=V])>

L'agente SYNT richiede a MORPH di cercare la frase nominale del verbo *like* [le ambiguità potrebbero essere: *Pilots, planes, can* sono verbi (to pilot/to plane) o nomi (a pilot/a plane/a can of beer); *like* è anche verbo (to like) o preposizione (like); flying è anche un gerundio (to like + gerundio) o un aggettivo di planes]:

H4: <SYNT><MORPH><asynchrone><request><propose><P2 (S1) ><seek-SN=C1)>  
 H4. <MORPH><SYNT><asynchrone><request><propose><P2 (S1) ><flying (gerond) planes (F (noun) ) >  
 H3. <SYNT><MORPH><asynchrone><answer><confirm><P2(S2)><ph([like=V])>

MORPH usa l'informazione I3 per proporre *can* come nome.

<sup>6</sup> N.H. Minsky. The imposition of protocols over open distributed systems. IEEE Transactions on Software Engineering, February 1991.

H5: <MORPH><SYNT><asynchrone><request><propose> <P2(S2)><ph([can=F] )>  
H5 <SYNT><MORPH><asynchrone><ansuer><modify> <P2 (S1) ><ph( [can=V] ) >  
H3: <MORPH><SYNT><asynchrone><request><propose> <P2(S2)><ph( [like=P] )>  
H3 <SYNT><MORPH><asynchrone><request><confirm> <P2 (S1) ><ph( [like=P]) >  
H3 <SYNT><all><asynchrone><inform><accept> <P2(S5)><ph( [like=P])>

Il sistema si ferma quando non ci sono più aggiornamenti di interpretazioni da fare.

# APPROCCIO LESSICO-STRUTTURALE VS APPROCCIO LINGUISTICO-COGNITIVO (1996/1997) [3]

Molti sistemi nell'ambito dell'NLP sono progettati come sistemi puramente sequenziali che hanno notevoli limiti di performance.

L'obiettivo di molti studi avuti in questi ultimi anni è la creazione di sistemi distribuiti come alternativa all'approccio tradizionale dei sistemi di NLP dal momento che questi possono offrire cooperazione fra moduli autonomi, specializzati e distribuiti.

L'idea di considerare agenti come unità di processo e conoscenza in grado di usare il loro potere di comunicazione per ottenere un risultato è molto allettante.

Comunque sembrano esserci almeno due differenti possibilità per distribuire la conoscenza linguistica fra gli agenti:

3. una **distribuzione lessico-strutturale**: in questo caso gli agenti sono associati alle parole di una frase rispettando legami di unione di tipo grammaticale. Questo approccio, Chomskiano per natura, rivela una possibilità di studiare le conseguenze di ipotesi di unione che portano verso una rappresentazione strutturale della frase
4. una **distribuzione linguistico-cognitiva**: qui il MAS contiene agenti più potenti associati ai tradizionali livelli di processamento linguistico(morfologico, sintattico e semantico)<sup>7</sup> e qualche volta con fenomeni linguistici quali anafora<sup>8</sup>, ellissi<sup>9</sup>, coordinazione, ambiguità di categoria.

In generale quando associamo agenti ad ogni livello linguistico abbiamo pochi agenti, se paragonati a quelli che avremmo associando agenti per ogni parola di una frase, ma molto più complessi.

---

<sup>7</sup> Linguistica è la disciplina scientifica che studia :

- a. la lingua intesa come potenziale innato dell'uomo di produrre linguaggio. In altre parole, la linguistica studia la capacità espressiva umana pre-esistente alla sua realizzazione completa. Fino a circa due anni di vita il neonato possiede la lingua ma non ancora il linguaggio.
- a. il linguaggio in quanto prodotto di questa facoltà. Linguaggio sono ad esempio italiano, inglese o il linguaggio dei segni o altri sistemi comunicativi umani complessi, costituiti cioè di segmenti minimi portatori di significato articolati fra loro in un sistema gerarchico complesso(es periodo).

La linguistica generale si occupa solo di elaborare le categorie ed i concetti con cui descrivere tale innata capacità. Essa si può suddividere nelle aree di:

- fonologia, morfologia e sintassi che insieme formano quella che tradizionalmente è chiamata grammatica
- metrica, semantica, pragmatica e lessicologia.

<sup>8</sup> Anafora è una figura retorica che consiste nella ripetizione di una parola(o più) per sottolineare un'immagine o un concetto(esempio, per me si va nella città dolente, per me si va nell'eterno dolore...)

<sup>9</sup> Ellissi è una figura retorica che consiste nell'omissione, all'interno di una frase, di uno o più termini che sia possibile sottintendere. E' frequente nei proverbi e nelle scienze. Esempio: "A nemico che fugge ponti d'oro". E' usata anche in matematica sotto forma dei tre puntini "..."

Sono state analizzate entrambe le distribuzioni.

Per testare la distribuzione lessico-strutturale è stato implementato un sistema dove gli agenti sono associati alle categorie morfologiche come verbo, nome, aggettivo e così via [7, bibliografia citata]. Un'agente cerca, nel suo vicinato, un altro agente che possa completarlo o che sia da questo completato per formare strutture sempre più grandi come ad esempio una frase nominale(NPs), una frase aggettivale(AdjPs), una frase preposizionale(PPs), etc. Il procedimento continua fino a che non vengono trovate strutture sempre più complesse.

Per testare la distribuzione linguistico-cognitiva è stato creato un sistema dove gli agenti sono associati a livelli linguistici [8, bibliografia citata].

In ogni caso entrambi i progetti sono stati realizzati per il Portoghese anche se lo sviluppo futuro è quello di estendere tale applicazione a tutte le altre lingue.

Prima di analizzare entrambi i sistemi vediamo in generale quali sono le caratteristiche essenziali di agenti in generale e nel particolare di Sistemi Multi Agente.

Un agente è un'entità intelligente capace di agire, comprendere la conoscenza di un ambiente perseguendo i suoi obiettivi(goals) in un modo razionale e intenzionale in base al suo stato attuale di conoscenza(stato interno) che, però, potrebbe cambiare attraverso la cooperazione con altri agenti nello stesso ambiente.

Dobbiamo, però, considerare i seguenti aspetti:

- capacità, conoscenza, goals relativi ad un dominio linguistico.
- comunicazione esplicita fra agenti con un linguaggio comune.
- percezione dell'ambiente e descrizioni esterne su altri agenti.
- capacità di agire.
- controllo sotto attività individuale e cooperativa.

Le attività individuali degli agenti sono determinate da un comportamento autonomo che è raggiunto attraverso la percezione di cambiamenti nell'ambiente che un agente riesce a percepire tramite l'uso di adeguati protocolli di comunicazione.

In base al loro tipo di comportamento abbiamo due tipi di agenti:

1. **Agenti Reattivi** il cui comportamento è basato su risposte a stimoli e le azioni ed i comportamenti degli altri agenti sono percepiti come cambiamenti nell'ambiente.
2. **Agenti Cognitivi** che sono capaci di ragionare sulle azioni degli altri agenti in quanto possiedono una rappresentazione esplicita dell'ambiente e degli agenti nella società.

In un sistema Multi-Agente gli agenti devono interagire con altri agenti in primo luogo per acquisire quanta più conoscenza possibile che permette loro di raggiungere un obiettivo.

Un grosso vincolo è che gli agenti devono interagire con incerte ed, alle volte, anche contraddittorie sorgenti di informazioni.

In ogni caso bisogna anche considerare il fatto che un agente non sarà in grado, tutte le volte, di ricavare tutte le possibili soluzioni al problema, per il raggiungimento del proprio obiettivo, ma solo alcune di esse. In questo caso gli agenti dipendono da capacità di ragionamento e interazione con l'ambiente in primo luogo per aumentare la propria conoscenza. Da qui, ma anche per risolvere possibili situazioni di conflitto di goal o informazione, è necessario considerare gli agenti come entità che interagiscono fra loro.

Vediamo adesso più nel dettaglio i due tipi di distribuzione esistenti.

## Approccio di distribuzione lessico-strutturale

L'articolo da cui trae ispirazione il sistema realizzato, basato su un approccio di distribuzione lessico-strutturale, è il [7, bibliografia citata].

In questo approccio ogni agente è associato ad una differente categoria morfo-sintattica come per esempio nome, verbo, articolo determinativo, etc. Una parola è considerata una unità di conoscenza ed il significato di una frase è ottenuta attraverso l'interazione fra agenti; questi agenti tentano di trovare altri agenti con cui poter stabilire associazioni formando, così, strutture sempre più grandi fino a che l'intera struttura della frase non può essere rappresentata. Queste associazioni non possono essere considerate causali, bensì sono regolate da **forze di attrazione** e **meccanismi di repulsione/restrizione**.

L'idea alla base delle **forze di attrazione** è rappresentata dalle forze di influenza che alcune parole esercitano su altre. Tali forze sono determinate dalla categoria della parola nel modo seguente: i verbi attraggono nomi, preposizioni e pronomi relativi; i nomi attraggono preposizioni e articoli determinativi; le preposizioni attraggono i nomi; i pronomi relativi attraggono i nomi e le preposizioni; le congiunzioni attraggono i verbi ed i nomi.

Vengono usati tre differenti **meccanismi di repulsione/restrizione**: *sottocategorizzazione* (solo per verbi), *vicinato* e *restrizione selezionata*.

La *sottocategorizzazione* determina la categoria di argomenti che possono riempire una certa posizione nella matrice dei verbi. Così per esempio, non è abbastanza dire che un verbo può essere seguito da una preposizione; bisogna specificare il tipo di complemento semantico della preposizione.

*Vicinato* è il meccanismo che indica il tipo di influenza di una parola in termini di nozione spaziale e può essere visto come un meccanismo debole di checking sintattico: è necessario che un agente abbia un altro agente nel suo vicinato.

*Restrizione selezionata* specifica le possibili combinazioni di significati che possono occorrere fra parole e può essere applicata per eliminare combinazioni incoerenti.

Detto questo è facile capire come il processo d'interpretazione di una frase può essere visto come la ricerca di parole che possono modificare o essere modificate da altre in base alle forze di attrazione e vincolate dai meccanismi di restrizione.

E' necessario però creare una forte assunzione iniziale: si assume che l'ambiguità di appartenenza ad una data categoria di una parola (come ad esempio accade quando una parola può appartenere a più categorie, per esempio nome e verbo) sia già risolta in una fase di pre-processamento. In questo modo se una parola è associata ad un agente di un tipo specifico (basata sulla categoria della parola) assumiamo che questa parola possa avere più significati ma tutti all'interno della medesima categoria morfologica.

L'interazione fra agenti durante il processo di ricerca avviene attraverso lo scambio di messaggi di rete o diretti.

Gli agenti nominali giocano un ruolo importante nell'interpretazione di una frase anche se questi, all'inizio del processo, si ritrovano in uno stato di attesa passiva in attesa che qualche agente verbale, o di altro tipo, modifichi il suo stato. Gli agenti che possono modificare lo stato di un agente nominale sono solo quelli presenti nel suo vicinato. La fase di congiunzione degli agenti inizia quando un agente ne trova un altro che pensa di poter modificare, chiedendo l'aggancio. La fase termina quando tutti gli agenti sono collegati ad altri.

Le ragioni per cui gli agenti non possono attaccarsi/congiungersi sono:

- frase mal formata

- la frase è ben formata ma c'è stata un'azione errata da parte di qualche agente
- la frase è formata da una parola

Il sistema è capace di elaborare l'esistenza della *topicalizzazione*, delle *preposizioni relative*, di *mancanze in una frase(gapping)* e delle *congiunzioni coordinative*(e, anche, pure, eppure, nè, ..).

La *topicalizzazione* è lo spostamento di qualche componente della frase all'inizio della stessa anche se quella non è la sua posizione reale.

Uno dei problemi con le *preposizioni relative* è la distanza che potrebbe esistere fra verbo e soggetto; questa lontananza potrebbe essere notevole e potrebbe comportare un notevole carico computazionale. Questo problema è stato elegantemente risolto dal sistema di Paiva [7, bibliografia citata] dal momento che è l'agente verbale che invia messaggi cercando elementi corrispondenti.

Quando invece vengono trattate situazioni in cui sono presenti gap o congiunzioni coordinative è difficile ricostruire la frase di partenza.

Quando un agente nominale trova sè stesso, termina il suo naturale comportamento passivo ed inizia ad esplorare il vicinato cercando un verbo o una preposizione. Questo cambio di comportamento è la soluzione al problema del gapping. La ricostruzione è fatta creando un nuovo agente verbale che opererà esattamente come specificato nel paragrafo precedente. Quando invece non trova un verbo o una preposizione nel suo vicinato inizia a cercare una congiunzione coordinativa(ad esempio *e*); fatto ciò continua con il cercare un verbo nella frase precedente. Se questo è il caso viene creato un nuovo agente verbale da inserire nella frase mal formata. Questo nuovo agente ricomincerà la ricerca ed alla fine tutti gli agenti saranno collegati fra loro e la frase sarà considerata ben formata.

## Approccio di distribuzione linguistico-cognitiva

Questo approccio è basato su [2] anche se differisce da quest'ultimo per almeno due importanti aspetti: (1) Stefanini [9, bibliografia citata] non usa una base di conoscenza semantica; (2) il suo modello include agenti con compiti di pre-processamento che non esistono nel modello di Silva [8, bibliografia citata] dal momento che nel sistema esistono agenti morfologico-lessicali.

Il modello ad agente linguistico include la conoscenza linguistica ed un set di procedure per manipolarla con lo scopo di raggiungere il proprio obiettivo. Il modello è composto da:

- dominio - capacità, conoscenza e goal di quest'agente (analisi morfologico-lessicale, sintattica e semantica, risoluzione di anafore, etc.) ed in grado di accedere allo stato interno degli agenti.
- capacità di comunicazione - protocollo di cooperazione
- descrizioni esterne - informazione sulle capacità di altri agenti(che hanno meccanismi di ragionamento sociale).

Per quanto riguarda l'implementazione di agenti linguistici, esistono due livelli: (1) dominio linguistico proprio considerato per il problema della decomposizione in NLP (i livelli linguistici sono considerati unità di agentificazione); (2) **livello di cooperazione**, composto da moduli di comunicazione e da descrizioni esterne, che sono replicati in ogni agente nella società.

## Dominio Linguistico

In questo livello, con il termine *capacità* di un agente si intende l'esecuzione di analisi morfologico-lessicale, di parsing, analisi semantica. Grammatiche, dizionari e reti semantico-lessicali sono altri attributi principali della specialità di ogni agente.

### **Livello di Cooperazione**

Il Livello di Cooperazione consiste in moduli di comunicazione e ragionamento sociale. Lo scambio di un messaggio consiste su una comunicazione diretta fra agenti. Ogni agente possiede una mailbox per inviare e ricevere messaggi in modo asincrono. Viene usato il tradizionale meccanismo di scambio di messaggi <SENDER, MESSAGE, RECEIVER>, con in più qualche primitiva presa dal linguaggio di comunicazione **KQML** per rappresentare intenzioni e cooperazione.

Una delle proprietà di un agente è quella di riconoscere gli altri agenti nella società. Questa capacità sociale è rappresentata come un "modello di un altro", una struttura dati che associa goals, struttura, identificazione di altri agenti nella società e che è specifica nella descrizione esterna di agenti.

### **Confronto dei due Sistemi**

Il confronto dei due sistemi sopra descritti avviene in relazione ai seguenti aspetti:

- organizzazione degli agenti
- costruzione delle società e sotto-società
- altre possibili agentificazioni
- distribuzione di conoscenza
- composizione del dizionario
- aumento del sistema

### **Organizzazione degli agenti e trattamento della frase**

La prima grande differenza sta nell'identificazione degli agenti nei due diversi modelli: nel modello di Paiva [7, bibliografia citata] gli agenti corrispondono a categorie di parole mentre nel modello di Silva [8, bibliografia citata] gli agenti sono associati a livelli linguistici.

Per come è stato definito il modello di Paiva, quindi, gli agenti associati con categorie di parole sono duplicati a causa del fatto che il MAS potrebbe aver bisogno di tutte le parole che compongono la frase.

Ad esempio volendo analizzare la frase "*the house and the garden*" abbiamo bisogno di due agenti-determinativi, due agenti-nome ed uno congiunzione.

Questa duplicazione, che deriva dalla frase che si vuole analizzare, sembra essere una caratteristica intrinseca del sistema MAS che fa una decomposizione spaziale del problema.

Per come, invece, è stato definito il modello di Silva non c'è duplicazione di agenti(sebbene questa caratteristica si nota analizzando testo e non singole frasi). Gli agenti associati a livelli di processamento linguistico(morfologico-lessicale, sintattico, semantico) sono organizzati in un modo che assomiglia fortemente ad un approccio sequenziale. L'analisi della frase inizia con un trattamento morfologico-lessicale attraverso l'agente morfologico che, una volta terminata la fase di processamento, invia il risultato all'agente sintattico per la costruzione degli alberi di derivazione. Quest'ultimo invia, per finire, il risultato ottenuto all'agente semantico per la

costruzione della struttura semantica. L'aspetto negativo è che per una semplice frase quale "*il gatto miagola*" il sistema usa una successione di trattamenti morfologici - sintattici - semantici.

### **Costruzione della sotto società**

Nel modello di Paiva le sotto-società sono costruite dinamicamente; il processo è controllato da forze di attrazione e meccanismi di repulsione. E' possibile affermare che è proprio la costruzione della società a garantire il trattamento della frase. Ad esempio per la frase "*the house and the garden*" viene creata un'interazione/società fra l'agente nome e quello determinativo formando la frase nominale "*the house*". Lo stesso avviene per la frase nominale "*the garden*". Queste sotto-società continuano a crescere fino a quando non coprono l'intera frase.

Anche nel modello di Silva la creazione delle sotto-società è un'operazione dinamica. La differenza è che queste sono costruite per risolvere fenomeni linguistici e non per regolare associazioni strutturali regolari. Ad esempio non appena viene trovata un'ambiguità (che potrebbe causare la generazione di strutture di rappresentazione differenti per la stessa frase) a livello sintattico, viene creata una sotto-società composta da agenti sintattici e semantici.

Ciò nonostante, a causa del piccolo numero di agenti nel modello, possono essere create poche possibili organizzazioni. In una frase come "*I sing it*" gli agenti interagiscono formando una sotto-società sintattico-semantica per risolvere la seguente ambiguità: *sing* è un nome o un verbo?

### **Possibili altre agentificazioni**

Nel modello di Paiva l'organizzazione interna degli agenti, che include la conoscenza circa le forze di attrazione e i meccanismi di repulsione, potrebbe dare luogo all'utilizzo di due agenti per ognuna di queste forze opposte. Questo tipo di realizzazione prevede, però, una diversa rappresentazione delle sorgenti di conoscenza che potrebbero essere utilizzate per configurazioni molto più sofisticate di questa.

Nel modello di Silva c'è la possibilità di ulteriore agentificazione proprio nell'agente sintattico anche se questa modifica porterebbe ad avere un agente molto più sofisticato.

### **Conoscenza distribuita fra agenti**

Nel modello di Paiva gli agenti sono ragionevolmente semplici e somiglianti ad agenti reattivi. Alla base contengono conoscenza grammaticale. In altre parole la distribuzione di conoscenza è basata su un criterio strutturale di possibili associazioni in base alle classi di parole esistenti. Le conoscenze morfologiche e semantiche sono contenute in un dizionario che deve essere completo in termini di conoscenza ed ampliato in termini di entità per parole del linguaggio; questa conoscenza è esterna agli agenti.

Nel modello di Silva la conoscenza dichiarativa e procedurale, diffusa fra gli agenti, chiaramente riflette la natura linguistico-cognitiva degli agenti morfologici-lessicali, sintattici e semantici.

In questo modello gli agenti contengono dizionari e analizzatori relativi ad ognuna di queste fasi. Per esempio l'agente morfologico contiene un dizionario morfologico-lessicale che consente la decomposizione di parole o espressioni ed un analizzatore morfologico.

L'agente sintattico contiene una grammatica, cioè un dizionario sintattico ed il corrispondente analizzatore sintattico.

L'agente semantico contiene un dizionario semantico basato sulla semantica del lessico generativo di James Pustejovski [1] ed i relativi meccanismi generativi proposti in questa teoria

### **Composizione del dizionario**

I dizionari sembrano essere fondamentali per il successo di un modello che tratta frasi del linguaggio naturale.

Nel modello di Paiva il dizionario è una struttura esterna al modello che viene consultata non appena la decomposizione di una frase e le associazioni pertinenti diventano possibili.

Nel sistema di Silva, invece, è adottata una strategia completamente differente.

Il dizionario, così come le procedure di accesso allo stesso, sono distribuite fra gli agenti. Ogni agente di un livello linguistico, quindi, contiene un dizionario e le rispettive procedure di accesso. In questo sistema le strutture vengono distribuite in base al tipo di conoscenza necessaria all'agente. Il numero di voci, in ognuno di questi dizionari, è varia. Per esempio il dizionario sintattico contiene voci per ogni sottoinsieme del dizionario morfologico. Proprio a causa di questa distribuzione di conoscenza, il processo di aggiornamento(esterno agli agenti) è particolarmente importante. E' richiesta coerenza in primo luogo per garantire integrità di informazione.

### **Comunicazione di risultati parziali**

Nel sistema di Paiva i risultati parziali sono associati ad una struttura sintattica (un frame) costruita gradualmente. Questo frame possiede una voce per ogni unità lessicale trattata.

Nel sistema di Silva esiste un protocollo di comunicazione, implementato usando primitive KQML.

Il protocollo di conoscenza cooperativa si basa su una comunicazione fra agenti attraverso una rete usando primitive basate su azioni della lingua parlata. L'esistenza di un protocollo in questo modello è cruciale e permette eterogeneità agli agenti.

### **Crescita del sistema**

Per come è stato progettato il modello di Paiva non è prevista alcuna crescita/aumento del numero di agenti nel sistema a causa di una copertura totale fra numero di agenti e categorie di parole esistenti in una frase.

Nel sistema di Silva, invece, possono aggiungersi agenti in qualunque momento.

# HERALD: UN APPROCCIO ROBUSTO AL NLP (2001) [\[4\]](#)

Nonostante le abilità e le conoscenze dell'essere umano derivano da tanti anni di sviluppo culturale, anche quest'ultimo può non essere in grado di interpretare correttamente il linguaggio naturale qualora non abbia opportuna conoscenza del contesto e del dominio del discorso o l'appropriata competenza del lessico usato. In realtà è possibile dare un'interpretazione, sbagliata o giusta che sia, ad ogni tipo di discorso(o più in generale dato) anche se questa potrebbe poi non essere la migliore esistente. Se tutto questo accade per gli essere umani, figuriamoci per le macchine!

La reale differenza è che l'essere umano potrebbe anche non avere conoscenza di un settore specifico, e quindi non comprendere la giusta interpretazione della frase, riuscendo però a colmare le lacune tramite le sue capacità di apprendimento.

NLP (o Processamento del Linguaggio Naturale non ristretto) è un compito dell'AI fra quelli ritenuti difficili(hard).

Recenti tecniche, denominate “tecniche robuste”, assumono che l'analisi perfetta del problema non esiste, ma che esistono analisi/risultati parziali del tutto soddisfacenti. Per riuscire a gestire questo tipo di tecniche bisogna prendere in forte considerazione problemi come la mancanza di conoscenza, l'incertezza, la vaghezza delle informazioni, l'ambiguità ed i malintesi.

In un sistema ritenuto robusto, tale qualità dev'essere raggiunta assemblando tutti i componenti in un unico sistema globale facendo sì che ognuno di essi riesca a fornire la migliore soluzione possibile ad un dato sotto-problema considerato.

Un aspetto particolarmente interessante è la **modularità** applicata al NLP.

Con il termine modularità si intende la possibilità di non basare l'analisi secondo una rigida decomposizione del linguaggio in componenti linguistici(per esempio sintassi, semantica e pragmatica), bensì di applicare autonomi moduli di analisi che a turno possono considerare diversi aspetti del linguaggio.

La modularità ma in particolare l'indipendenza modulare(con cui i diversi moduli di analisi operano) rappresenta un problema chiave: affinché si riesca a fornire una modellazione globale del linguaggio è necessario che i diversi moduli autonomi di analisi siano composti da meta-regole globali. Per ottenere questo goal è necessario che tali moduli siano forniti di un linguaggio comune di comunicazione in primo luogo per garantire accessibilità globale.

I principali vantaggi nel considerare la modularità come base di analisi per NLP sono:

- l'estrazione di caratteristiche del linguaggio a diverse granularità(fenomeni, parole, frasi, discorsi..)
- l'esistenza di differenti strategie di analisi che incarnano differenti principi

- la copertura è ottenuta come composizione(ma anche sovrapposizione) di differenti modelli e tecniche di analisi

Esistono diversi sistemi distribuiti basati su un approccio cooperativo al NLP.

**TREVI** [2, bibliografia citata] è un sistema supportato da un ambiente per applicazioni NLP basate sull'object-oriented in cui i moduli cooperano sotto il controllo di un oggetto manager(data-flow based).

**Incremental Discourse Parser architecture** [3, bibliografia citata] prevede uno schema di coordinazione non predefinito. Questo sistema appartiene ai cosiddetti sistemi a "lavagna"<sup>10</sup> che forniscono un framework dove differenti moduli esperti cooperano per la risoluzione di un problema. Tali moduli controllano costantemente la lavagna in attesa di messaggi per loro. Il coordinatore tipicamente controlla l'accesso alla lavagna non consentendo una computazione veramente distribuita.

**TALISMAN** [2] è un'agenzia dove ogni agente ha una specifica competenza linguistica. Gli agenti scambiano informazioni fra di loro utilizzando un apposito linguaggio di interazione.

Ogni agente è governato da un proprio set di regole(leggi) interno che determina il comportamento specifico dello stesso.

Un set di regole globale definisce invece la comunicazione(coordinazione).

Di fronte ad un problema di disambiguazione semantica, per esempio, un agente chiede il supporto a qualche altro agente esperto. Questi agenti negoziano fra loro in caso di conflitto nella risoluzione al problema.

**CARMEL** [4, bibliografia citata] è forse il più famoso Sistema Multi-Agente.

Tale sistema combina due differenti livelli di processamento che lavorano in parallelo: livello/processo subliminale e livello/processo conscio.

Tali livelli sono Sistemi Multi-Agente differenti con principi diversi.

**PROCESSO SUBLIMINALE**: è un processo che lavora in secondo piano e spesso lasciato non supervisionato. E' composto da moduli che lavorano su input scritto su una struttura dati più evoluta rispetto alla lavagna(*blackboard*) chiamata *sketchboard*. Tali moduli operano in cascata in modo che ogni modulo fornisce il suo output come input per quello successivo.

Non esiste output del sistema fino a quando l'input non è stato processato a dovere, e cioè non subisce più alcuna modifica.

Il processo subliminale (o nascosto) è collegato con il processo conscio (o visibile) fornendo a questo il suo output come risultato finale della computazione sotto forma di pop-up.

Tale evento scatena processi controllati che incapsulano procedure linguistiche esperte.

Il processo conscio è costruito come un Sistema Multi-Agente.

L'architettura prevede un meta-sistema dominato da meta-agenti ognuno dei quali controlla un dominio specifico in cui agiscono agenti specifici. Tale dominio specifico rappresenta a sua volta un meta-sistema e così via.

---

<sup>10</sup> Blackboard (lavagna) è una struttura dati centralizzata governata da un coordinatore

Un meta-agente incapsula conoscenze aggiuntive per coordinare più agenti diversi con compiti diversi.

Tutti questi sistemi hanno come caratteristica in comune quella di avere, come base teorica, un efficiente algoritmo fra i molti che esistono come strategia di analisi e di risoluzione al NLP.

Ognuno di questi algoritmi efficienti ha, però, una serie cospicua di parametri a scelta libera da impostare a priori. La scelta di questi influenza notevolmente l'efficienza dell'algoritmo.

Le motivazioni per cui si sceglie un Sistema Multi-Agente per una robusta analisi intelligente della lingua sono le seguenti:

- *sorgenti d'informazione distribuita*: le sorgenti linguistiche e di conoscenza possono derivare da fonti molto diverse (anche luoghi fisicamente diversi). L'Accesso a tali sorgenti multiple potrebbe essere reso in un modo uniforme.
- *condivisibilità*: le applicazioni accedono ai servizi ed alle risorse in modo totalmente asincrono per adempiere compiti differenti. Sarebbe inutile e dispendioso replicare tutte le risorse per ogni applicazione esistente. Viceversa sarebbe molto produttivo consentire un accesso condiviso da tutti gli agenti per il recupero di informazioni o risorse.
- *nascondere la complessità*: un'architettura distribuita permette di definire più livelli indipendenti per la risoluzione di problemi in cui i dettagli di coordinazione sono nascosti ai livelli più astratti.
- *modularità e riusabilità*: un tema chiave nello sviluppo di una robusta applicazione di un sistema di analisi è l'integrazione di applicazioni autonome esistenti. Gli Agenti, utili ai nostri scopi, potrebbero essere inglobati in applicazioni linguistiche pre-esistenti. Proprio per risolvere tale problema i linguaggi di comunicazione fra agenti garantiscono interoperabilità fra servizi eterogenei.
- *flessibilità*: gli Agenti Software devono adattare il loro comportamento a nuove possibili configurazioni che cambiano nel tempo, seguendo il flusso di informazioni o il cambiamento di requisiti.
- *robustezza*: quando informazione e controllo sono distribuiti, il sistema è capace di agire efficacemente anche quando un agente non è in grado di fornire il suo servizio. Questo è particolarmente importante per sistemi di NLP in quanto possono esistere imprevedibili e nascosti fenomeni.
- *qualità di informazione*: l'esistenza di similari modelli di analisi permette di avere interpretazioni multiple dello stesso input. Questo può essere molto interessante in quanto offre una possibilità di garantire la correttezza dei dati attraverso validazioni incrociate e la possibilità di valutare la migliore interpretazione offerta.

## The HERALD project

Un sistema modellato utilizzando gli Agenti Cognitivi sembra essere molto appropriato per il NLP in quanto permette di creare un sistema dove ogni agente potrebbe costruirsi e manipolare la propria base di conoscenza utilizzandola per prendere decisioni.

Per il progetto HERALD è stato preso in considerazione un modello più semplice, AOP Agent Oriented Programming, dove la progettazione degli agenti è fatta utilizzando l'impiego di regole.

**AOP è un framework realizzato appositamente come una specializzazione della programmazione orientata agli oggetti, dove questi ultimi diventano agenti con la ridefinizione del loro stato interno e dei protocolli di comunicazione.**

Laddove i semplici oggetti contengono valori arbitrari come stato e comunicano con messaggi verosimilmente non strutturati, i più complessi agenti contengono credenze, obiettivi, scelte e comunicano rispettando protocolli che contengono atti linguistici come informare, chiedere, promettere, declinare. Lo stato di un agente è esplicitamente definito come uno *stato mentale*.

Per adattare AOP per la progettazione di sistemi ad agenti di NLP sono state riviste alcune caratteristiche:

- *Gestione dello stato mentale in ViewGen*: ViewGen è un sistema che modella le caratteristiche di un agente. Un ambiente rappresenta ogni agente; più ambienti corrispondono a più agenti. Gli ambienti possono interagire fra di essi. Ad ogni ambiente è associato un sistema di ragionamento. ViewGen è utilizzato per meglio rappresentare, in modo più sofisticato, la gestione dello stato mentale di un agente.
- *Capacità di intermediazione*: esiste un agente mediatore il cui compito è quello di prelevare ed associare le capacità di risoluzione di differenti agenti per specifici sotto-problemi, tramite primitive KQML. Ogni agente specifica le sue competenze per mezzo di un *Capability Description Language (CDL)*.
- *IRC*: International Relay Chat è il server di chat più usato in internet. E' stato accorpato al progetto in quanto permette agli agenti di spedire messaggi in ambienti distribuiti grazie ad una rete di infrastruttura basata su tecnologia peer-to-peer.

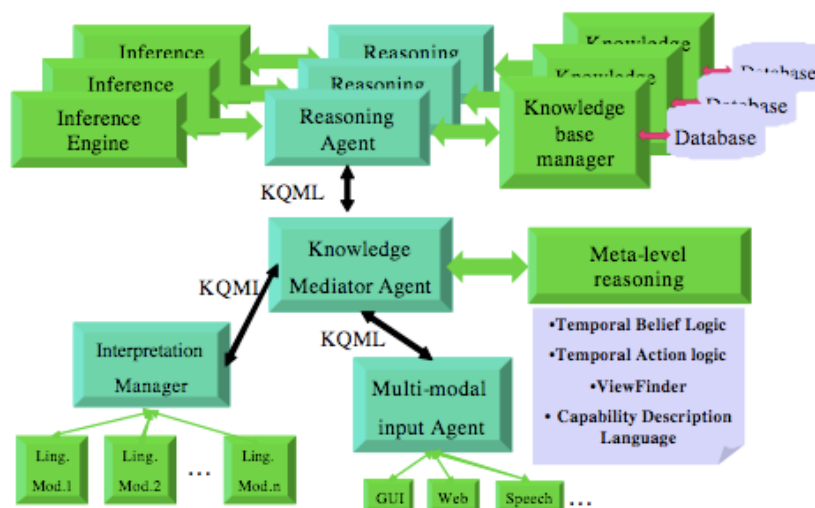
Più nello specifico vediamo quali sono i temi principali della ricerca/sviluppo di questo framework robusto per il NLP.

**Estendere la copertura.** Passando da un'architettura classica sequenziale ad una distribuita il problema della copertura è rilevante.

**Migliorare l'efficienza.** Un'architettura distribuita ci permette di lanciare istanze multiple dello stesso parser con parametri e vincoli differenti. In questo modo sarà possibile valutare e confrontare tutti i risultati ottenuti.

**Disambiguazione.** E' forse uno degli aspetti più importanti e difficili nel NLP. Un'interpretazione incrementale permette di risolvere problemi di ambiguità usando informazioni contestuali accumulate a diversi livelli linguistici e attraverso un *filtering semantico guidato dal dominio di competenza*. Questa tecnica ci permette di escludere involontarie interpretazioni usando opportune basi di conoscenza generali o specifiche (esempio ontologie esistenti incapsulate negli agenti).

Nel seguito riportiamo l'architettura generale di HERALD per applicazioni NLP:



**Fig. 2.** The HERALD general architecture for NLP applications

Il *Reasoning Agent* ingloba un motore di inferenza<sup>11</sup> ed una base locale di conoscenza.

Il *Multi-Modal input Agent* rappresenta un'interfaccia di interazione fra le varie forme di input ed il sistema.

L' *Interpretation Manager* si occupa della corretta interpretazione dell'input fornito.

Il *Knowledge Mediator Agent* è l'agente responsabile della coordinazione fra i moduli/agenti diversi presenti nel sistema. Il suo compito primario è quello di assegnare i vari compiti ai diversi agenti esperti appartenenti al sistema, secondo alcune politiche di gestione appropriate. Il KM-Agent recupera le risorse disponibili e pianifica il corso delle azioni in base alle capacità degli agenti reclutati.

In HERALD sono presenti alcuni formalismi, introdotti per la progettazione del framework per AOP:

*Temporal Belief Logic* [6]: permette di tenere traccia dell'evoluzione dello stato mentale di un agente. Questo è utile per due motivi: in primo luogo per verificare formalmente le proprietà del sistema multi agente ed in secondo luogo per fornire un meta-linguaggio adatto per la comunicazione del comportamento del sistema multi-agente nel complesso.

*Capability Description Language*: l'utilità di considerare un meccanismo con capacità di intermediazione è chiaro se vediamo il nostro sistema come un'architettura aperta dove i moduli possono essere aggiunti senza nessun intervento.

*View Finder - Viewpoints Logic*: è un framework per la manipolazione di ambienti/domini. Un ambiente/dominio fornisce una netta demarcazione del confine di informazione fornendo benefici metodologici e di processo (ragionamenti definiti, etc.).

<sup>11</sup> Il motore inferenziale e' un programma costituito da un interprete che decide quale regola applicare per poter aumentare la base di conoscenza e da uno scheduler che organizza le regole da sviluppare e il loro ordine di esecuzione. Il compito del motore è estrarre le regole utili alla soluzione del problema secondo un meccanismo di riconoscimento e attivazione delle stesse, eseguendo un esame delle regole nella base di conoscenza, selezionare la regola più appropriata, eseguire al regola e registrare nella memoria di lavoro l'azione, ripetendo finchè non trova la possibile soluzione. Esso è organizzato in due parti : una memoria di lavoro (o lavagna) dove viene memorizzato il piano generale di soluzione ed un'agenda delle cose ancora da fare, oltre ad una descrizione della soluzione o delle soluzioni per ora ipotizzate.

In HERALD è stato testato anche il riconoscimento e l'interpretazione del testo (*text understanding*) che richiede un'intersezione fra l'informazione linguistica e la conoscenza del mondo. Tale integrazione potrebbe essere richiesta ad ogni livello di descrizione linguistica.

GETARUNS(General Text And Reference Understanding System)<sup>12</sup> [7] è un sistema esistente per l'analisi e l'interpretazione testuale.

Nonostante sia un sistema del tutto generale, trova forte utilizzo nel trattamento di testo narrativo.

Durante la fase di processamento vengono riconosciute e trattate tutte le relazioni e le entità presenti nel testo in base alla loro rilevanza.

GETARUNS è formato dai seguenti moduli linguistici:

- *parser sintattico di tipo top-down*
- *modulo trattamento anafora a livello di frase*
- *modulo di risoluzione anafora a livello di discorso*
- *moduli semantici per la generazione di forme logiche e la raccolta di quantificatori*
- *moduli di analisi temporale a livello di frase*
- *moduli di analisi temporale a livello di discorso*
- *moduli di ragionamento e rappresentazione concettuale*
- *modulo di costruzione del modello del mondo*

GETARUNS è concettualmente un'architettura modulare anche se è implementata come un programma PROLOG monolitico e sequenziale.

Come sviluppo futuro è prevista una reingegnerizzazione di GETARUNS con un approccio ad Agenti ma nel frattempo si provvederà ad adattare l'analisi di novelle testuali con il sistema HERALD proponendo euristiche per la modifica del flusso di dati e dei parametri di processamento.

---

<sup>12</sup> <http://project.cgm.unive.it/html/sharedtask/> o <http://project.cgm.unive.it/html/Risorse/GetarunIndex.html>

# KQML: UN ESEMPIO DI LINGUAGGIO COORDINATIVO

[5]

Risulta particolarmente difficile immaginarsi, soprattutto se si parla di sistemi multi agente, un ambiente in cui le entità esistenti siano isolate.

Risulta, invece, logico immaginarsi un ambiente in cui tutte le entità siano strettamente correlate le une con le altre.

Da ciò deriva l'idea che un'entità risulta essere un agente software se e solo se comunica con un opportuno linguaggio di comunicazione ad agenti.

Da qui nasce l'importanza di definire, nel modo opportuno, un linguaggio di comunicazione che riesca a garantire l'interazione *intelligente, costruttiva e significativa* fra agenti software.

Una caratteristica fondamentale, senza cui non avrebbe senso parlare di corretta interazione fra agenti, è la capacità di capire e riconoscere il significato dei termini usati in una frase in relazione al loro contesto di appartenenza. Per esempio parliamo di *velocità*: ciò che un fisico potrebbe conoscere parlando del concetto di velocità è diverso da quanto potrebbe intendere un appassionato di automobili, ipotizzando che si voglia intavolare un discorso sulle macchine veloci. Da tutto ciò deriva l'importanza di associare ad ogni parola un'etichetta, come risultato del processo di comprensione della stessa in base al suo dominio di appartenenza, parzialmente codificata nel linguaggio (questa operazione è possibile solo a patto di rispettare alcuni pattern di comunicazione del tipo domanda-risposta).

In modo del tutto analogo, quando la comunicazione e l'interazione avviene fra agenti software sono necessari tre componenti distinti:

- *un linguaggio di comunicazione*
- *un modello di comprensione comune sulla conoscenza condivisa*
- *abilità nel condividere quanto detto nei punti precedenti*

Queste caratteristiche possono essere radunate con il termine di *interoperabilità* che diventa molto diffusa nei sistemi computazionali di oggi.

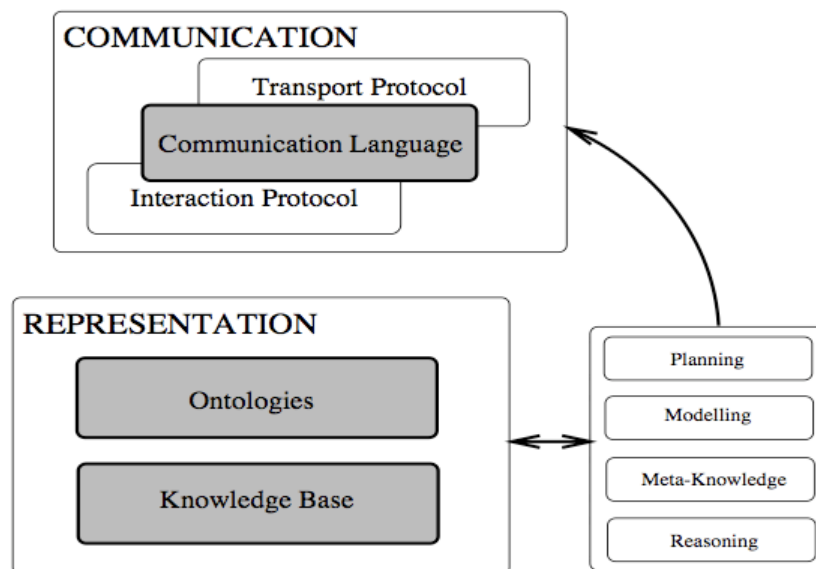
L'interoperabilità è il fondamento dell'approccio del consorzio **KSE** (**K**nowledge **S**haring **E**ffort).

L'approccio che andremo ad approfondire si basa sul linguaggio di comunicazione del KSE e sul protocollo **KQML** (**K**nowledge **Q**uery and **M**anipulation **L**anguage).

## KSE - Knowledge Sharing Effort

Un agente software può essere visto come un'applicazione per cui l'abilità di comunicare e scambiare conoscenza con altre applicazioni è di importanza primaria.

La Figura sottostante chiarisce come dovrebbe essere composto un agente che abbia le caratteristiche di cui sopra:



La Figura mostra un modello astratto di Agenti Software interagenti che sono identificati da tre classi di componenti: componenti per la rappresentazione, componenti per la comunicazione e componenti che non sono direttamente collegati con lo scambio di conoscenza

Lo scambio di conoscenza può essere diviso in due sottoproblemi: 1. traslazione da un linguaggio di rappresentazione della conoscenza ad un altro(o da una famiglia di linguaggi di rappresentazione di conoscenza ad altre); 2. condivisione di contenuti semantici di conoscenza fra applicazioni differenti.

**La sola traslazione di conoscenza non è sufficiente per risolvere tale problema in quanto ogni base di conoscenza trattiene implicite assunzioni sul significato di cosa è rappresentato. Se due applicazioni volessero semplicemente traslare le informazioni, si perderebbero parte delle conoscenze implicite che invece devono essere trasferite. In sintesi, il contenuto semantico di ogni parola, o termine, dev'essere preservato.**

La comunicazione è un triplice problema che necessita della conoscenza di un *protocollo d'interazione*, un *linguaggio di comunicazione* ed un *protocollo di trasferimento*.

Il **protocollo d'interazione** si riferisce ad una strategia di alto livello seguita da un agente software che vuole comunicare con altri agenti. In quanto protocollo può essere usato come schema di negoziazione o semplicemente come strumento per chiedere a qualcuno qualcosa.

Il **linguaggio di comunicazione** è il mezzo attraverso cui scambiare il contenuto della comunicazione. E' il linguaggio di comunicazione che suggerisce se il contenuto della comunicazione è un'asserzione, una richiesta o qualche forma di domanda.

Il **protocollo di trasporto** è il meccanismo di trasporto usato per la comunicazione come TCP, HTTP, etc..

KSE è un'iniziativa per sviluppare un'infrastruttura tecnica di supporto allo scambio di conoscenza fra i sistemi.

E' organizzata in tre gruppi di lavoro ognuno dei quali si occupa di una parte specifica: *Interlingua*, *Shared Reusable Knowledge Bases*, *External Interfaces*.

*Interlingua Group*: sta sviluppando un linguaggio comune per esprimere il contenuto di conoscenza. Questo gruppo ha pubblicato un documento specifico che descrive **KIF**(Knowledge Interchange Format)<sup>13</sup>, uno strumento che può essere usato come supporto per la traslazione da un linguaggio ad un altro fra agenti che usano differenti linguaggi nativi di rappresentazione.

*SRKB(Shared Reusable Knowledge Bases) Group*: si occupa di conoscenza condivisa e scambio della conoscenza con la creazione di tools indipendenti dal contesto<sup>14</sup>. Il risultato ottenuto è chiamato **Ontolingua**.

*External Interface Group*: si occupa di interazione a tempo d'esecuzione fra sistemi basati sulla conoscenza ed altri sistemi che si evolvono nel tempo. Particolare attenzione è stata data a due importanti casi: comunicazione fra due sistemi di conoscenza e comunicazione fra sistema di conoscenza e sistema di gestione convenzionale di database. **KQML** è il linguaggio creato da tale gruppo e rappresenta il loro maggior risultato.

## KIF

KIF è la soluzione suggerita da KSE per gli aspetti sintattici di rappresentazione per lo scambio di conoscenza. Tale linguaggio è l'espressione di conoscenza e meta-conoscenza.

Esistono due motivazioni che spiegano la creazione di tale linguaggio:

- la creazione di una *lingua franca* per lo sviluppo di applicazioni intelligenti, con particolare enfasi sulla cooperazione.
- la creazione di un formato comune di interscambio utilizzato per la traduzione da un linguaggio A a KIF e da KIF ad un linguaggio B (invece che la traduzione diretta da A a B).

## Ontologie e Ontolingua

SRKB ha concentrato i suoi sforzi di ricerca sul problema di condivisione del contenuto di conoscenza(semantica) formalmente rappresentato e nella creazione di un software che può essere riutilizzato in ambiti diversi.

---

<sup>13</sup> Ulteriori informazioni su KIF e tools associati sono reperibili sul sito <http://www.cs.umbc.edu/kse/kif>

<sup>14</sup> Ulteriori informazioni sul sito web <http://www.cs.umbc.edu/kse/srkb>

Il gruppo ha lavorato per la creazione di ontologie<sup>15</sup> in vari domini.

Ogni ontologia è scritta utilizzando KIF e definisce un insieme di classi, funzioni, costanti per qualche discorso includendo anche assiomi per vincolare l'interpretazione del testo garantendo la disambiguazione.

## KQML

E' concepito come formato di messaggi e come protocollo di scambio di messaggi in sistemi runtime formati da Agenti.

Ecco le principali caratteristiche:

- i messaggi sono oscurati, non contenendo meramente le frasi da comunicare, ma l'attitudine delle frasi inerente al contesto(asserzione, richiesta, domanda).
- un ambiente, in cui la comunicazione è gestita attraverso l'uso del KQML, può contenere agenti chiamati *facilitatori* poichè provvedono a creare un'associazione fra indirizzi fisici e nomi simbolici, registrazione di databases e servizi cercati e/o offerti da agenti, servizi di comunicazione(inoltro, intermediazione, etc.). Giusto per usare una metafora per spiegare meglio la funzionalità dei facilitatori, essi sono i segretari degli agenti nel loro dominio.

Vediamo più nel dettaglio le caratteristiche di KQML.

Affinchè sia possibile la comunicazione fra agenti è necessario che questi usino un linguaggio comune o, in alternativa, linguaggi che possono essere tradotti.

Ma anche ipotizzando che una di queste due ipotesi sia realizzata, è ancora necessario, affinché la comunicazione fra agenti avvenga in modo corretto, che sia condiviso un framework di conoscenza per poter interpretare i messaggi scambiati. Proprio per questo motivo non si tratta di uno scambio di semantiche bensì di ontologia. Lo scambio di ontologia è una procedura molto utilizzata soprattutto in domini quali pianificazione, schedulazione, biologia e medicina.

KQML è un linguaggio ed un set di protocolli che supportano gli agenti software nell'identificare, connettere e scambiare informazioni con altri agenti software.

Esso è formato da tre livelli:

- lo *strato di contenuto* in cui troviamo il contenuto del messaggio
- il *livello di comunicazione* dove troviamo codificato un set di caratteristiche del messaggio che descrivono i parametri di comunicazione ad un livello basso(ad esempio l'identità del mittente)
- il *livello di messaggio* che codifica il cuore vero e proprio del messaggio, contenendo informazioni sul tipo di interazione fra agenti(il protocollo usato, il tipo di messaggio, il tipo di linguaggio, l'ontologia specifica).

La sintassi di KQML è fondata sull'utilizzo di parentesi bilanciate. L'elemento iniziale della lista rappresenta il tipo di messaggio, mentre tutti gli altri elementi sono argomenti formati da parola chiave - valore.

Vediamo un esempio di comunicazione fra *joe* e *IBM* circa il prezzo di vendita di una quota di magazzino:

---

<sup>15</sup> Ulteriori informazioni sul sito web <http://www-ksl.stanford.edu/knowledge-sharing/ontologies/README.html>

(ask-one

**:sender joe**  
**:content (PRICE IBM ?price)**  
**:receiver stock-server**  
**:replay-with ibm-stock**  
**:language LPROLOG**  
**:ontology NYSE-TICKS)**

In questo messaggio la performativa KQML è ask-one (una richiesta) il cui contenuto è (PRICE IBM ?price), l'ontologia è NYSE-TICKS, il ricevente è un server di nome stock-server, il linguaggio con cui è scritta la query è chiamato LPROLOG.

Volendo suddividere la query sopra, riportata come esempio, in livelli abbiamo che il **:content** è lo strato di contenuto; **:replay-with**, **:sender**, **:receiver** il livello di comunicazione; **:language**, **:ontology** il livello di messaggio.

Un'ipotetica risposta alla query di cui sopra, potrebbe essere il seguente:

(tell

**:sender stock-server**  
**:content (PRICE IBM 14)**  
**:receiver joe**  
**:in-replay-to ibm-stock**  
**:language LPROLOG**  
**:ontology NYSE-TICKS)**

Un elenco di tutti i possibili tipi di messaggio esistenti(performative) è riportato nel grafico sottostante:

| <i>Category</i>               | <i>Name</i>                                      |
|-------------------------------|--------------------------------------------------|
| <b>Basic query</b>            | evaluate, ask-if, ask-about, ask-one, ask-all    |
| <b>Multi-response (query)</b> | stream-about, stream-all, eos                    |
| <b>Response</b>               | reply, sorry                                     |
| <b>Generic informational</b>  | tell, achieve, cancel, untell, unachieve         |
| <b>Generator</b>              | standby, ready, next, rest, discard, generator   |
| <b>Capability-definition</b>  | advertise, subscribe, monitor, import, export    |
| <b>Networking</b>             | register, unregister, forward, broadcast, route, |

Ad esempio il tipo *advertise* indica ad agente che sta per ricevere un avviso da parte di un altro agente, oppure *recruit* che è usato per reclutare agenti disponibili per un particolare compito e per ricevere un determinato tipo di messaggio.

Un esempio pratico potrebbe essere il seguente:

(advertise

**:ontology NYSE-TICKS**

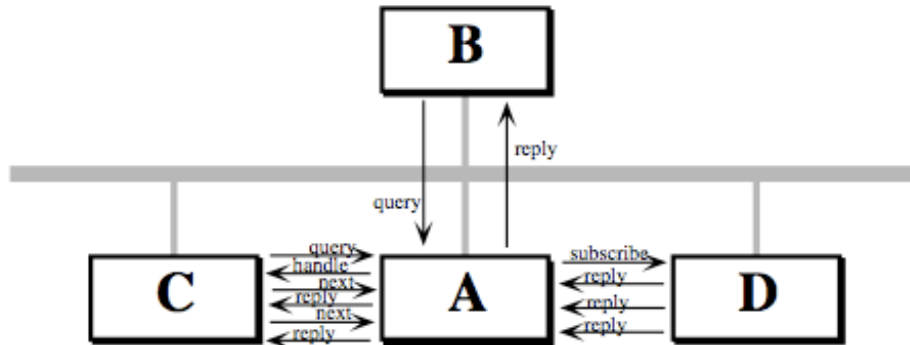
```

:language LPROLOG
:content (stream-all
          :content (PRICE ?x ?y)))

```

Il server che riceve tale messaggio riceverà un messaggio di richiesta specifica per una particolare operazione (richiesta economica).

In KQML esistono diversi tipi di protocolli.



Analizziamo i diversi protocolli schematizzati in figura sopra.

Caso fra gli agenti A e B: un agente agisce come client ed invia una query ad un altro agente che invece rappresenta il server, aspettando la risposta. Il server risponde con una singola risposta o un set di risposte.

Caso fra agenti A e C: la risposta del server non è completa ma è solo una parte introduttiva che indica al client di rimanere in attesa dei rimanenti pezzi di risposta. Questo è il tipico caso in cui viene richiesto un database relazionale o quando è prevista una sequenza di istanziazioni in risposta. Sebbene questo tipo di scambio prevede che il server mantenga uno stato interno, le singole transazioni individuali sono come nel caso precedente, instaurando una comunicazione sincrona fra agenti.

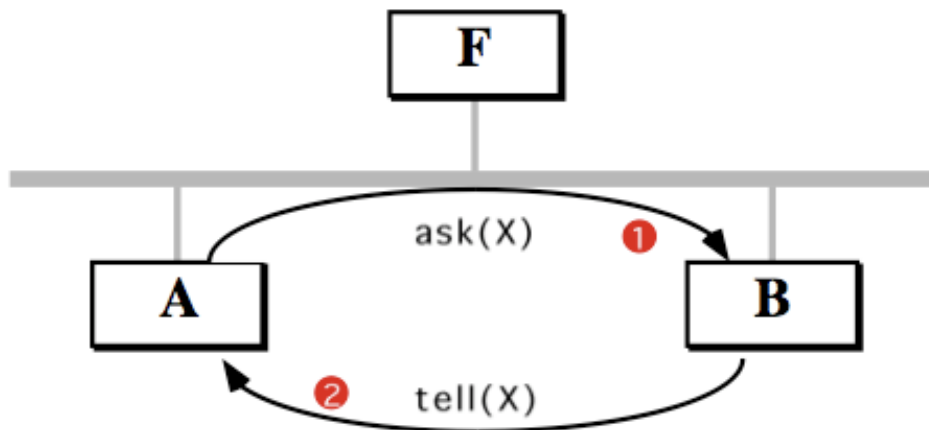
Caso fra agenti A e D: il client sottoscrive una richiesta al server che invia un indefinito numero di risposte asincrone che arrivano ad intervalli irregolari. La comunicazione è, ovviamente, asincrona e all'arrivo di una risposta il client potrebbe anche essere occupato in altre comunicazioni.

In realtà potrebbero esistere delle varianti a tali schemi: ad esempio potrebbe accadere che i messaggi non vengano inoltrati ad uno specifico indirizzo server ma vengano inoltrati nella rete ad un numero di agenti. Le risposte, che potrebbero arrivare in maniera sincrona o non, devono essere raccolte e associate alla query di cui esse sono la risposta.

Lo scopo, nel creare KQML, è stato quello di creare un supporto a differenti architetture ad agenti. Proprio grazie alla notevole varietà di architetture esistenti, è nata l'idea di introdurre agenti chiamati facilitatori della comunicazione (*communication facilitator*).

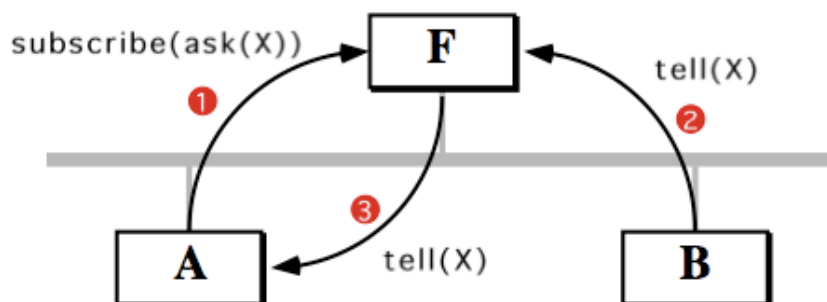
Un facilitatore è un agente che esegue diversi tipi di servizi di comunicazione (per esempio mantiene un registro dei nomi dei servizi, inoltra i messaggi per un dato servizio, distribuisce i messaggi in base al contenuto).

Come esempio prendiamo il caso in cui un agente A avrebbe piacere di conoscere la verità di una frase X, ed un agente B ha nella sua base di conoscenza proprio la frase X. In questa comunicazione è disponibile un agente facilitatore F.

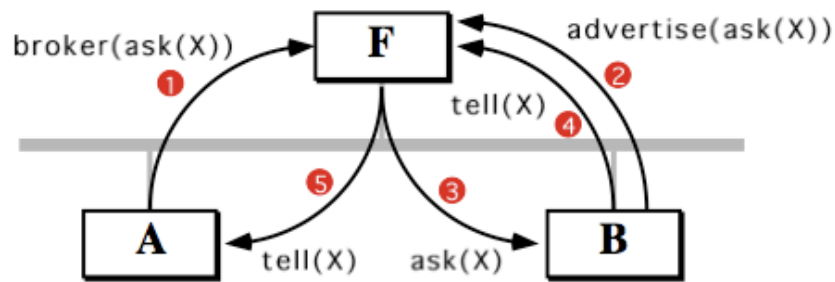


Se A è consapevole che sia appropriato inviare una richiesta di X a B, allora può usare un protocollo di comunicazione punto-punto ed inviare la query direttamente a B, come in figura sopra.

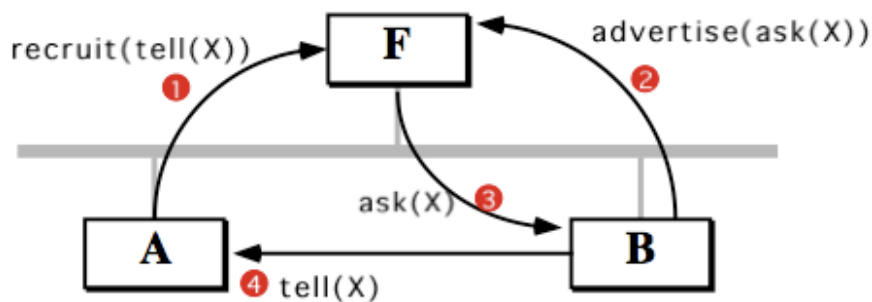
Se, invece, A non è consapevole di quali agenti sono disponibili, o di quale agente ha X nella sua base di conoscenza, o come contattare un agente fra quelli disponibili, allora può usare una serie di approcci.



In figura sopra è schematizzato un primo tipo di approccio: A usa la performativa *subscribe* per sottoscrivere una richiesta al facilitatore F riguardo la verità di una frase X. B successivamente informa F del risultato ed F ritorna tale valore ad A.

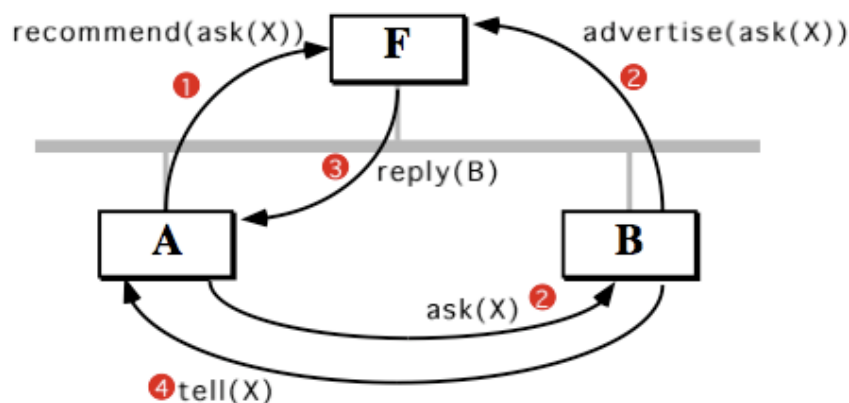


In figura sopra è rappresentata una situazione leggermente diversa: A chiede ad F di cercare un agente che può processare la performativa *ask* (*broker*). B avverte F, in maniera del tutto indipendente, che è disponibile per accettare performative del tipo *ask*. Una volta che F ha entrambi questi messaggi, inoltra a B la richiesta inviata da A, aspetta risposta da B ed inoltra quest'ultima al richiedente A.



In figura sopra, A usa una performativa leggermente diversa, rispetto al caso precedente, per informare F del suo interesse riguardo la conoscenza della verità su una frase X.

La performativa *recruit* consente di chiedere di ricercare un agente in grado di processare la richiesta ed inviare la risposta direttamente a mittente.



Come caso finale prendiamo in considerazione quello in figura sopra.

A chiede ad F di consigliare un agente in grado di rispondere alla sua richiesta. Una volta che F è venuto a conoscenza delle potenzialità di B (che risulta in grado di rispondere a domande su X), segnala ad A il nome dell'agente B. A questo punto A è libero di instaurare un rapporto di comunicazione direttamente con B.

Da tutto ciò riusciamo a capire quanto è importante il ruolo di un agente facilitatore.

Come un agente riesce a trovare un facilitatore non è un compito di competenza del KQML.

Analizziamo brevemente quali sono le caratteristiche del linguaggio per come lo abbiamo descritto in questa sezione.

**FORMA:** la forma di descrizione delle primitive del linguaggio sembra essere una forma dichiarativa. Il formato è una sequenza lineare di caratteri. Sebbene questo formato è irrilevante per le funzioni del linguaggio esso permette di rendere i messaggi particolarmente chiari, facili da leggere, da convertire in altri formati, da trasformare. La semplice sintassi permette di aggiungere nuovi parametri in future revisioni.

**CONTENUTO:** come già affermato, il linguaggio KQML prevede tre livelli. Il livello del contenuto, quello del messaggio e quello di comunicazione. I messaggi KQML sono ignari del contenuto che essi trasportano. Per quanto riguarda il livello di messaggio, esistono diverse performative che rappresentano atti/gesti del parlato. Il livello di comunicazione permette di fissare alcuni parametri di basso livello come ad esempio l'identità del mittente o quella del destinatario.

**SEMANTICA:** è ancora un campo di ricerca aperto. E' prevista una maggiore enfattizzazione della lingua parlata negli atti di comunicazione presenti nel linguaggio KQML.

**RETE:** gli agenti possono comunicare direttamente con altri agenti, inviare i loro messaggi nella rete o appoggiarsi all'uso di facilitatori per assolvere il loro compito. KQML permette sia la comunicazione sincrona che quella asincrona.

**AMBIENTE:** KQML può utilizzare qualunque protocollo come meccanismo di trasporto(HTTP; TCP/IP, ...). Inoltre poichè il contenuto dei messaggi è oscurato, non è imposto alcun limite al tipo di contenuto da trasmettere. E' garantita interoperabilità con altri linguaggi di comunicazione.

**AFFIDABILITA':** non sono previsti moduli di sicurezza o autenticazione. Il contenuto dei messaggi è oscurato ma non è prevista nessuna correzione dei messaggi o recupero messaggi smarriti. Sono

previsti dei tipi di messaggi usati nei casi in cui un'applicazione non può processare o comprendere il contenuto(*error* e *sorry*).

## **Ontologie e Ontolingua**

SRKB ha concentrato i suoi sforzi di ricerca sul problema di condivisione del contenuto di conoscenza formalmente rappresentato e nella creazione di un software che può essere riutilizzato in ambiti diversi.

Il gruppo ha lavorato per la creazione di ontologie in vari domini.

Ogni ontologia è scritta utilizzando KIF e definisce un insieme di classi, funzioni, costanti per qualche discorso includendo anche assiomi per vincolare l'interpretazione del testo garantendo la disambiguazione.

# Symposium: Software Agents with Natural Language Capabilities - Where we are?

(2003)

[6]

BRIMS 2003: In questa occasione è stata enfatizzata l'importanza dello sviluppo di Agenti Software nel trattamento del Natural Language Processing avendo come base di studio le capacità del linguaggio parlato.

Si ritiene che, in questo settore, non si potrà parlare di Agenti Software completamente formati e strutturati fino a quando questi non saranno in grado di comunicare con altri Agenti o con esseri umani secondo i termini ed il linguaggio umano (da qui l'esigenza di introdurre KQML come ricerca su un linguaggio di coordinazione fra agenti).

Qui nel seguito, verrà riportata una breve trattazione sullo stato attuale dell'arte dando maggior peso a quei progetti, fino ad adesso realizzati, che hanno sviluppato Agenti Software nell'ambito del NLP con capacità di linguaggio parlato.

Rispetto al passato, in cui si era sottostimato la difficoltà di creare sistemi software che operassero nel settore del NLP, lo sviluppo teorico in tale direzione è stato garantito da uno sviluppo tecnologico di notevole rilevanza: la dimensione di memoria, la potenza computazionale, gli sviluppi nella teoria Linguistica e nella teoria Cognitiva, lo sviluppo di notevoli basi di conoscenza strutturata e di [corpora annotati](#) (APPENDICE A), il riconoscimento vocale e i sistemi di sintesi.

Alcuni sviluppi teorici e tecnici hanno portato miglioramenti di prospettive per lo sviluppo di sistemi di agenti software con NLP e capacità di riconoscimento della lingua parlata:

- l'avvento della [Linguistica Cognitiva](#) [1] con il suo focus sul significato in contrapposizione alla Sintassi Generativa basata sulle forme linguistiche più che sul significato dei termini.
- il successo dello sviluppo nel riconoscimento vocale e nelle tecnologie di sintesi
- lo sviluppo e la disponibilità di utilizzo di evoluti sistemi di conoscenza (esempio reti semantiche) che rappresentano un supporto per lo studio del NLP (esempio WordNet<sup>16</sup>, CYC<sup>17</sup>, FrameNet<sup>18</sup>)
- lo sviluppo di evolute tecniche statistiche per supportare la comprensione ad un livello più basso, esempio Latent Semantic Analysis

---

<sup>16</sup> WordNet è una rete semantico-lessicale basata sul concetto di synset cioè raggruppamento di termini con eguale significato (sinonimia). Pagina Web di riferimento: <http://wordnet.princeton.edu/>

<sup>17</sup> CYC è un progetto il cui scopo è quello di assemblare un'ontologia comprensiva ed una base di conoscenza che rappresentasse una conoscenza di "senso comune" con lo scopo di supportare applicazioni basate sul ragionamento umano. Pagina Web di riferimento: <http://www.cyc.com/>

<sup>18</sup> FrameNet è una risorsa lessicale on line con lo scopo di sviluppare combinazioni semantiche e di sintattiche di ogni parola in ogni suo senso comune. Pagina Web di riferimento: [http://framenet.icsi.berkeley.edu/index.php?option=com\\_frontpage&Itemid=1](http://framenet.icsi.berkeley.edu/index.php?option=com_frontpage&Itemid=1)

- lo sviluppo di ambienti virtuali a di interfacce che supportano la creazione di agenti software interagenti

Sebbene ci siano stati, quindi, notevoli passi avanti nello sviluppo di tecnologie nel NLP e nel riconoscimento della lingua parlata, esistono tuttora notevoli barriere da superare prima di considerare gli Agenti Software con capacità di NLP come un sistema/approccio stabile al problema del NLP:

- la piena integrazione di tecnologie nell'ambito del NLP
- il trattamento di ambiguità strutturali e lessicali del linguaggio
- la modellazione del contesto di uso del linguaggio
- il trattamento di significati non letterali delle parole
- la manipolazione di sorgenti non ben strutturate dal punto di vista grammaticale
- la manipolazione di sorgenti degradate
- la gestione di notevole mole di materiale e conoscenza linguistica
- l'integrazione di realizzazioni teoriche e tecnologiche provenienti da più discipline in un sistema coerente

Avendo come conoscenza preliminare il quadro finora presentato, esistono notevoli progetti sviluppati nel campo di Sistemi Multi Agente applicati al NLP con capacità di interazione utilizzando la lingua parlata.

Nel seguito riportiamo una breve descrizione dei principali progetti secondo uno schema che prevede, per ogni progetto, di sottolineare quali sono stati i principali sviluppi teorici/tecnologici che lo hanno influenzato, quali sono state le principali barriere teoriche/tecnologiche e quali le principali barriere politiche(esempio mancanza di fondi).

### **The AWO Tutor (Bruce Roberts, BBN Technologies)**

Il sistema AWO(Air Weapons Officer) Tutor è responsabile nell'avvertire, monitorare e dirigere le azioni dei piloti durante una missione in aerei AWACS<sup>19</sup>.

In effetti l'uso di AWO mantiene la consapevolezza dei piloti in quanto tutta la comunicazione avviene con il parlato.

In un sistema AWO un utente assume il ruolo di operatore su un aereo e comunica con altri operatori, su altri aerei, iniziando un rapporto di domanda-risposta. Gli altri operatori possono essere umani o Agenti Software AWO in grado di comunicare. Un agente AWO(tutor) osserva l'interazione fra AWO ed i piloti per stabilire la qualità della comunicazione e della comprensione o per suggerire azioni, in tempo reale, all'Agente AWO.

I sistemi AWO combinano riconoscimento della lingua parlata e sistemi di sintesi con tecnologia di parsing. E' un sistema malleabile grazie alla sua enorme capacità di adattamento in sistemi con capacità estese di NLP.

Come sviluppo futuro c'è quello di rendere più efficienti le comunicazioni fra Agenti AWO e persone riducendo la lunghezza delle frasi.

---

<sup>19</sup> AWACS è un sistema elettronico di controllo e di allarme aviotrasportato. E' basato su un radar progettato per eseguire missioni di sorveglianza aerea e funzioni 3C(comando, controllo e comunicazione) sia per scopi tattici che per scopi di difesa. I sistemi AWACS possono comunicare con gli aerei amici estendendo il loro raggio d'azione(fino a 400 km di distanza) e dando loro ulteriore invisibilità radar in quanto questi non avranno più la necessità di usare i propri radar per rilevare gli attacchi nemici.

In conclusione il sistema AWO sembra essere uno dei pochi sistemi che include Agenti Software in grado di associare capacità di NLP con sintesi e riconoscimento della lingua parlata.

### **Mission Rehearsal Exercise (Jonathan Gratch, Institute for Creative Technologies)**

È un progetto basato sullo sviluppo di entità umane virtuali, Agenti Software autonomi in grado di compiere comunicazioni faccia-a-faccia in sistemi virtuali.

Finanziato dall'Esercito adesso e dalla Marina prima.

Gli Agenti Software virtuali comunicano, in questo mondo virtuale, con altri Agenti Software o con umani veri e propri esprimendo capacità di sintesi e di riconoscimento/comprendimento della lingua parlata, esprimendo emozioni visibili e compiendo azioni inerenti.

### **AutoTutor (Natalie Person & Art Graesser, Institute for Intelligence Systems)**

AutoTutor è un sistema web di tutoring intelligente basato su strategie pedagogiche e conversazione discorsiva.

Questa ricerca ha portato alla luce fondamenti empirici e teorici per la creazione di sistemi di supporto (di tutoraggio, tutoring) per gli studenti nell'apprendimento di conoscenza, instaurando conversazioni in linguaggio naturale su particolare temi.

Attualmente i temi sviluppati dal sistema AutoTutor sono l'informatica generale e la fisica Newtoniana. Ovviamente il sistema può essere ampliato con altri settori di competenza.

Il sistema si basa su un Agente che propone delle domande fra una serie di scelte.

Esiste anche un elenco di aiuti per sollecitare le conoscenze pre-acquisite da parte dello studente (in realtà tali aiuti sono presenti solo nelle versioni più ricche di contenuti).

L'Agente virtuale propone suggerimenti, aiuti, frasi con alcuni termini mancanti, identifica e corregge risposte errate e fraintendimenti nella domanda, risponde a domande proposte dagli studenti, crea e semplifica le risposte e propone dei punteggi come risultato dell'interazione che riflettono la preparazione e la qualità dell'interazione da parte dello studente.

In realtà quest'ultimo aspetto sembra avere una rilevanza fondamentale: a studenti che hanno maggior punteggio (secondo uno score fornito da pre-test) vengono proposti solo suggerimenti, mentre a studenti meno preparati (cioè studenti con score di pre-test basso) vengono fornite molte più informazioni da parte dell'Agente Software.

Questo tipo di Agente viene anche chiamato Agente Adattativo in quanto adatta la sua interazione ed il suo comportamento in base ad alcuni parametri, per esempio lo score di pre-test ottenuto dagli studenti.

### **Scott (Wayne Zachary, CHI System)**

Synthetic Cognition for Operational Team Training è un sistema che integra la pedagogia pratica guidata dall'intelligenza in sistemi interattivi con utenti virtuali.

L'ambiente di utilizzo di tale sistema può essere qualunque ambiente di difesa aerea spaziale dove le interazioni a voce sono il mezzo primario per stabilire e mantenere coordinazione fra le varie parti che compongono il sistema.

Nel sistema SCOTT E-2C gli agenti virtuali non solo hanno il compito di interagire parlando ma anche, e soprattutto, di ascoltare. L'ascolto è forse l'aspetto più importante perchè permette di acquisire nozioni importanti dalle affermazioni degli altri.

L'ascolto, però, aumenta il livello di difficoltà nella gestione degli agenti virtuali in quanto si aggiungono complessità temporali e di interazione: bisogna capire cosa sentire (alle volte può essere necessario ascoltare più fonti) o capire quando parlare (bisogna riconoscere i comandi imposti da un coordinatore che, per esempio, impone il silenzio sul canale di comunicazione) .

Da ogni comunicazione, l'Agente Software deve trarre le informazioni utili per operare in una situazione di azione tattico-operativa utilizzando un Agente Software appropriato di tipo cognitivo, quale COGNET/iGEN.

Il sistema SCOOT E-2C è integrato in altri sistemi quali STRATA.

## APPENDICE A – CORPUS

Un CORPUS o CORPORA è una raccolta di dati linguistici sotto forma di testi archiviati, al giorno d'oggi, in forma elettronica.

Un corpus non ha una grandezza predefinita a priori ma può variare dalle poche pagine di testo alle 400 milioni di parole della Bank of English.

I corpus vengono utilizzati in aree quali la ricerca linguistica, ricerca di informazioni, didattica, sviluppo di applicazioni per il trattamento del linguaggio naturale.

La struttura di base di un corpus prevede la specifica dei materiali che esso raccoglie, la codifica con cui il testo viene memorizzato, il sistema di interrogazione con cui vengono recuperate le informazioni ed il canale utilizzato per far arrivare i risultati al pubblico.

Un esempio:

<http://lettere.humnet.unipi.it/informaticaumanistica/ricerca> -> Corpus delle opere di Dante lemmatizzate

Materiali: opere di Dante

Codifica: XML - Tei<sup>20</sup>

Sistema di interrogazione: Interfaccia su motore di ricerca XCDE<sup>21</sup>

Canale per far arrivare i risultati a un pubblico: Internet

La differenza rispetto ad una semplice raccolta di testi sta nella facilità di eseguire diversi tipi di ricerche ed interrogazioni grazie ad una struttura dei materiali e degli strumenti appositamente realizzata.

Esempio: ricerca di una parola.

All'interno di un normale programma di scrittura si cerca una parola alla volta su un testo alla volta.

Un sistema di interrogazione su un corpus cercherà una parola all'interno di tutto il corpus, fornendo anche, e non solo, indicazioni statistiche.

---

<sup>20</sup> E' un tipo di codifica basati su documenti ben formati di tipo XML che seguono le regole inserite nella DTD. A causa della mancanza di specifiche opportune sul tipo di tag da utilizzare e sulla loro gestione, causando problemi di scarsa confrontabilità dei dati, vengono utilizzati dei marcatori più specifici o procedurali - TEI.

Ambizioni: fornire un formato standard per l'interscambio di informazioni; supportare la codifica di tutti i tipi di caratteristiche di ogni genere di testo.

<sup>21</sup> Motore di ricerca per il recupero di documenti, ed informazioni sugli stessi, codifica con XML -TEI (<http://www.di.unipi.it/~ferragin/Libraries/xcede/index.htm>). E' stato sviluppato presso il Dipartimento di Informatica, Università di Pisa.

# **APPENDICE B – LINGUAGGI DI PROGRAMMAZIONE PER LO SVILUPPO DI AGENTI**

## **Digressione sulla nascita e sull'importanza dei linguaggi di programmazione nei MAS**

Sebbene vi siano molte metodologie e strumenti per la realizzazione di agenti e di sistemi multi-agente, esse si possono inquadrare in due approcci di riferimento: vi sono modelli che impiegano la definizione di un agente e di un sistema multi-agente utilizzando tecniche di intelligenza artificiale e modelli che propongono un approccio più programmatico e sono maggiormente orientati allo sviluppo dei sistemi multi-agente proponendo framework per la programmazione di MAS. Un approccio complementare a quelli precedentemente presentati è rappresentato dall'adozione di un linguaggio di programmazione orientato agli agenti. Tale approccio è di fatto complementare ai precedenti in quanto molto spesso il linguaggio di programmazione orientato agli agenti si appoggia ad un framework ad agenti, che costituisce il target del codice sviluppato con il linguaggio. E' quindi l'accoppiamento del linguaggio di programmazione e del framework che permettono la realizzazione e la messa in opera di un'applicazione orientata agli agenti.

Lo sviluppo di sistemi ad agenti è stato affrontato nel tempo seguendo diversi approcci. Si è cercato infatti di costruire modelli che permettessero di definire delle specifiche formali del comportamento di sistemi ad agenti, ed allo stesso tempo sono stati forniti strumenti che permettessero la programmazione dei sistemi multi-agente. La realizzazione di specifiche formali di sistemi multi-agente è stata affrontata utilizzando modelli di programmazione logica di diversa natura. Gli strumenti per la programmazione dei sistemi multi-agente sono stati realizzati come pacchetti software sviluppati utilizzando la programmazione orientata agli oggetti. Questi pacchetti software sono normalmente conosciuti come framework per la programmazione ad agenti.

L'approccio basato sulla programmazione logica è maggiormente orientato alla definizione di un comportamento per l'agente e delle modalità di interazione che i diversi agenti instaurano; sono state sviluppate architetture che modellano la conoscenza che un agente ha del mondo, le sue intenzioni e gli obiettivi che esso intende perseguire che utilizzano come strumenti la logica dei predicati del primo ordine, la logica multimodale o altre soluzioni che costituiscono delle varianti a questi modelli. Soluzioni di questo tipo sono molto efficaci a livello espressivo ma spesso non permettono un'integrazione del tutto naturale con altri modelli di sviluppo software. L'approccio proposto con i framework per la programmazione di agenti, risulta in tal senso sicuramente più versatile ed è quello maggiormente utilizzato per lo sviluppo di sistemi multi-agente reali. Normalmente un framework per lo sviluppo di sistemi multi-agente propone un proprio modello di agente e di sistema multi-agente, offrendo una serie di strumenti e di interfacce semplificate che implementano processi automatizzati che guidano lo sviluppatore nella corretta definizione del

sistema e degli agenti. Queste sono soluzioni che aumentano la produttività, ma molto spesso le possibilità di intervento dello sviluppatore sono limitate sacrificando a volte l'efficacia di questo approccio. Spesso i sistemi multi-agente sviluppati attraverso i framework si appoggiano a una gerarchia di classi di un qualche linguaggio orientato agli oggetti (normalmente Java o C++) accessibile allo sviluppatore software. L'utilizzo diretto della gerarchia di classi offre sicuramente maggiori possibilità, ma si corre il rischio di perdere la prospettiva orientata agli agenti nello sviluppo del codice, soffermandosi su aspetti più tecnici della progettazione software, come ad esempio la gestione del threading. Inoltre seguendo questa strada spesso occorre ripensare in termini di oggetti l'applicazione originariamente progettata in termini di agenti. L'approccio basato sulla programmazione logica risulta adatto a definire l'intelligenza di un agente ma sicuramente non un agente software nel suo complesso, mentre l'approccio proposto con i framework offre un modello che permette di modellare efficacemente un sistema multi-agente ma può portare fuoristrada lo sviluppatore.

Il primo approccio allo sviluppo di applicazioni orientate agli agenti utilizzando un linguaggio di programmazione risale al 1991 con il linguaggio di programmazione Agent-0. I linguaggi di programmazione ad agenti permettono agli sviluppatori di definire e programmare agenti in accordo con un particolare modello di agente (quello supportato dal linguaggio). L'architettura BDI è il modello maggiormente implementato da questi linguaggi di programmazione. Tali linguaggi normalmente modellano gli agenti come componenti di un MAS, o come un insieme di asserzioni e clausole che debbono essere processate da un opportuno interprete. Sono stati perciò presi come modello per sviluppare i linguaggi di programmazione ad agenti due paradigmi di programmazione: il modello orientato agli oggetti e la programmazione logica. In particolare, la tecnologia Java è stata adottata come base per tutti quei linguaggi che hanno scelto il modello orientato agli oggetti come riferimento.

**Comunicazione fra Agenti: deve succedere per gli agenti quello che è successo per gli umani? Una miriade di linguaggi/dialetti diversi e incompatibili tra loro..**

Gli Agenti hanno una capacità di modificare l'ambiente, una conoscenza, un'intelligenza limitata. Per ovviare a questa limitazione, un Agente ha "**abilità sociale**" cioè capacità di interagire con degli altri Agenti per coordinare il loro operato e per negoziare risorse a attività. Il coordinamento permette: la risoluzione dei conflitti fra gli agenti; la realizzazione di compiti comuni. La negoziazione permette: la risoluzione dei conflitti fra gli Agenti; lo scambio di risorse e attività. L'abilità sociale di un Agente è possibile grazie ad un uso di un linguaggio di comunicazione.

### **Speech Acts Theory**

Framework concettuale per la comunicazione umana basato su teorie di uso del linguaggio (Austin 1962, "How to Do Things with Words").

Parlando si compiono atti comunicativi(*speech acts*) tipo *asserzioni*, *comandi*, *richieste*, *suggerimenti*, *promesse*, *minacce*, etc.. ma più nel particolare possiamo dire che ogni espressione parlata è un atto comunicativo(basta solo individuare di quale atto si tratta).

Quindi, un atto comunicativo trasmette conoscenza di qualche tipo: ma come fanno gli Agenti a comunicare? KQML.

La cosa importante è di specificare la semantica degli *speech act*. E' necessario che un Agente sia modellato come in possesso di una sua base di conoscenza(*KB*).

La standardizzazione nel mondo degli Agenti si chiama *FIPA Foundations for Intelligent Physical Agents* associazione senza scopo di lucro, fatta da università e imprese attive nell'ambito degli Agenti. La specifica è **FIPA ACL(1997)**.

## APPENDICE C – ONTOLOGIA

Nell'informatica, **un'ontologia** è il tentativo di formulare uno schema concettuale esaustivo e rigoroso nell'ambito di un dato **dominio**; si tratta generalmente di una **struttura dati** gerarchica che contiene tutte le entità rilevanti, le relazioni esistenti fra di esse, le regole, gli assiomi ed i vincoli specifici del dominio.

Un'**ontologia fondazionale** è in qualche modo assimilabile ad un glossario di base. Esempi di questo tipo sono il Basic English il dizionario delle 2000 parole della lingua inglese necessarie al dizionario necessario al dizionario Longman per descrivere le 4000 frasi più comuni della lingua inglese. Anche in Italia ci si è occupati di definire un italiano di base con gli stessi scopi e gli stessi metodi(Tullio De Mauro).

E' possibile anche un'analogia con i linguaggi artificiali. Tutti i programmi per computer si basano su ontologie fondazionali, costituite dall'insieme di istruzioni del processore, dalle librerie di un linguaggio, dai file presenti in un file system. Costruire la **rappresentazione di un dominio** di conoscenza partendo da delle basi insufficienti può portare a risultati poco corretti, da qui la necessità di disporre di ontologie di base standardizzate e consolidarle come fondamenta del proprio lavoro.

Malgrado il termine 'ontologia' sia stato utilizzato in modo estremamente generico per contraddistinguere un qualunque schema concettuale di classificazione, un'ontologia computazionale deve includere relazioni semantiche che descrivono in che modo i concetti sono interrelati.

Lo scopo di un'ontologia computazionale non è quello di specificare cosa 'esiste' o cosa 'non esiste', ma di creare una base di dati, che è un artefatto dell'uomo, contenente concetti riferiti ad un dominio e che verrà impiegata per eseguire certi tipi di computazione.

Le ontologie sono applicate comunemente nel campo dell'intelligenza artificiale e nella rappresentazione e nella condivisione della conoscenza. I programmi nei computer possono utilizzare un'ontologia per una varietà di scopi, fra cui il **ragionamento deduttivo**, la **classificazione**, diverse tecniche di **problem solving**, oltre che per facilitare la **comunicazione** e lo **scambio di informazioni fra diversi sistemi**.

Nessuna ontologia superiore è stata finora generalmente riconosciuta come standard **de facto** ma diverse organizzazioni stanno lavorando alla definizione di ontologie standard per domini di applicazione specifici(esempio il 'Process Specification Language' - PSL - creato dal National Institute for Standard and Technology - NIST).

Fra le ontologie disponibili elenchiamo:

- **Cyc** - <http://www.cyc.com> - è costituita da un'ontologia costitutiva(cioè non legata ad un particolare settore. E' chiamata anche ontologia superiore) e diverse ontologie di dominio. Un sottoinsieme di questa ontologia è stato rilasciato per uso libero col nome di **OpenCyc** (<http://opencyc.org>).
- **Wordnet** - <http://wordnet.princeton.edu> - è ampiamente usato nell'ambito del NLP. E' un database liberamente disponibile, progettato come rete semantica basata sui principi della psicolinguistica. Si qualifica come ontologia costitutiva perchè include sia i concetti di tipo generale che quelli con un maggior grado di specializzazione, collegati da relazioni semantiche.

- **SUMO**(Suggested Upper Merged Ontology) è un tentativo di definire un'ontologia superiore avviato dal gruppo IEEE. Disponibile per uso libero.
- **PLINIUS** - <http://cweb.inria.fr/Resources/ONTOLOGIES/AIMag18-03-006.pdf> - è un progetto sviluppato per tradurre sentenze scritte in linguaggio naturale in espressioni scritte utilizzando un linguaggio per rappresentare la conoscenza.
- **SENSUS** - <http://www.isi.edu/natural-language/resources/sensus.html> - è un'ontologia molto vasta che include sia termini di alto livello sia di livello intermedio.
- **ONTOWEB** - <http://www.ontoweb.org> - è una rete che unisce ricercatori e industriali potenziando le ontologie per migliorare lo scambio di informazioni nei seguenti campi: recupero delle informazioni, gestione della conoscenza, commercio elettronico e bioinformatica.

Fra i più comuni linguaggi per ontologie ricordiamo CycL(progetto Cyc), KIF, OWL(successore di RDF e RDFS), OIL e DAML(precedenti progetti di linguaggi per ontologie).

# BIBLIOGRAFIA CONSULTATA

- [1] PUSTEJOSKY J., *The Generative Lexicon*. The MIT Press, 1995.
- [2] STEFANINI M.H., DEMAZEAU Y., *TALISMAN: a multi-agent system for Natural Language Processing* in IEEE Conference on Advances in Artificial Intelligence, 12th Brazilian Symposium on AI, Campinas, Brasil, 1995, Springer, 310-320.
- [3] CARVALHO A.M.B.R., *Multi-Agent System for Natural Language Processing*, 1998.
- [4] PALLOTTA V., BALLIM A., *Agent-Oriented Language Engineering for Robust NLP*.
- [5] FININ T., LABROU Y., MAYFIELD J., *KQML as an agent communication language*, 1995.
- [6] BALL J.T., *Symposium: Software Agents with Natural Language Capabilities - Where are we?*, 2003.
- [7] KONING J.L., STEFANINI M.H., DEMAZEAU Y. *DAI Interaction Protocol as Control Strategies in a Natural Language Processing*, 1995.

# BIBLIOGRAFIA CITATA

- [1] PUSTEJOSKY J., *The Generative Lexicon*. The MIT Press, 1995.
- [2] BASILI R., MAZZUCHELLI L., PAZIENZA M.T., *An Adaptive and distributed Framework for Advanced IR* in 6th RIAO Conference (RIAO 2000), Paris 2000.
- [3] CRISTEA D., *An Incremental Discourse Parser Architecture* in Second International Conference - Natural Language Processing - NLP 2000, Patras, Greece, 2000, Springer.
- [4] SABAH G., *CARAMEL: a flexible model for interaction between the cognitive processes underlying natural language understanding*, 1990.
- [5] WOOLDRIDGE M.J., *On the Logical Modelling of Computational Multi-Agent System*, Department of Computation, UMIST, Manchester, UK, 1992.
- [6] DELMONETE R., *Parsing with GETARUNS*. in Proceeding of the “7eme conférence annuelle sur le Traitement Automatique des Langues Naturelles” TALN 2000, Lausanne, 2000.
- [7] PAIVA D.S.D., *Um Sistema Multi-Agentes para o Processamento Distribuído de Linguagem Natural*. Master Thesis, Universidade Estadual de Campinas, Brazil, 1996.
- [8] SILVA J.L.T.D., *Utilizacao do Paradigma Multi-Agentes no Processamento da Linguagem Natural: Um Modelo Voltado à Resolucao da Ambiguidade Lèxica Categorical na Língua Portuguesa*. Master Thesis, Pontificia Universidade Catolica do Rio Grande do Sul, Brazil, 1997.
- [9] STEFANINI M.H., *TALISMAN: un système multi-agent pour l’analyse du français écrit*, Thèse de Doctorat, Jan, 1993.