

Sistemi Multi-Agente LS

Approfondimento

Elaborazione di un modello per il manuale degli artefatti
a supporto del ragionamento pratico di agenti
intelligenti situati in ambienti computazionali secondo
il meta-modello A&A

Fabio Tonti
0000323634
fabio.tonti3@studio.unibo.it

Indice

1	Introduzione	5
2	Il Meta-Modello A&A	5
2.1	Nozione di Agente	6
2.2	Nozione di Artefatto	7
2.2.1	Il manuale	8
2.3	Nozione di Sistema Multi-Agente	9
2.4	Agenti pianificatori	10
3	Il planning	10
3.1	La pianificazione automatica: concetti base	11
3.2	La pianificazione classica	12
3.2.1	Planning Deduttivo	12
3.2.2	Planning mediante ricerca	12
3.2.3	Planning Gerarchico	14
3.3	Pianificazione e azione nel mondo reale	15
3.3.1	La pianificazione condizionale	16
3.3.2	La pianificazione reattiva	16
4	Planning e A&A	17
4.1	Dal planning al manuale	17
4.2	Ontologie	19
5	Modello del manuale	19
5.1	Grammatica del manuale	20
5.2	Esempi	24
6	Conclusioni e sviluppi futuri	26

1 Introduzione

Il meta-modello di programmazione Agenti e Artefatti (A&A) ha come obiettivo la modellazione e la progettazione di ‘working environments’ nel contesto di sistemi multi-agente cognitivi. Il working environment è un ambiente condiviso di lavoro, di cui fanno parte entità chiamate artefatti, che servono a regolare e supportare l’interazione tra gli agenti coinvolti. Ognuno di questi artefatti ha un suo manuale, che serve a descriverne le funzionalità; il manuale è destinato a essere esaminato e utilizzato a runtime da parte degli agenti, in modo da permettere a questi ultimi di scegliere gli artefatti e, in particolare, le operazioni più adatte per soddisfare i loro goal.

L’obiettivo di questo approfondimento è quello di proporre un modello per la descrizione del manuale degli artefatti, sfruttando possibili collegamenti con alcuni principi e concetti del planning tradizionale (AI Planning).

2 Il Meta-Modello A&A

Il meta-modello Agenti & Artefatti (A&A) è stato proposto come approccio general-purpose per ideare ambienti computazionali in Sistemi Multi-Agente (MAS), modellati come workspaces in cui gli agenti possono condividere, usare e manipolare gli artefatti. In un approccio di progettazione come questo, gli artefatti sono entità di prima classe (e quindi di particolare importanza), e rappresentano gli strumenti e le risorse che gli agenti possono sfruttare per portare a termine le loro attività individuali e sociali.

L’ispirazione dei concetti su cui si basa A&A deriva dalla Teoria dell’Attività (Activity Theory - AT), un approccio psicologico e concettuale nato nell’Unione Sovietica all’inizio del Ventesimo secolo. Uno dei principali concetti supportati dall’AT è che, nelle società umane, gli strumenti (cioè gli artefatti) giocano un ruolo fondamentale di mediazione nel fronteggiare il progressivo aumento della complessità nelle attività umane, in particolare quando si parla di attività sociali.

Per questo motivo, l’idea base di A&A è quella di introdurre la nozione di artefatto, come elemento di base per progettare quella parte del MAS che viene chiamata *working environment*. Questi elementi basilari possono essere programmati dai progettisti del MAS e istanziati, scoperti, condivisi, utilizzati e manipolati dinamicamente da parte degli agenti durante l’esecuzione delle loro attività. In altre parole, gli artefatti rappresentano la parte funzionale dell’ambiente del MAS, mentre gli agenti rappresentano la parte goal- o task-oriented. Diversi tipi di funzionalità possono essere incap-

sulate negli artefatti (programmati in modo appropriato): comunicazione e coordinazione, sincronismo, accesso a risorse, e così via.

Come strumenti di mediazione, gli artefatti possiedono funzioni sia di abilitazione, sia di limitazione: se da un lato estendono le abilità degli agenti per la manipolazione e trasformazione di diversi oggetti, dall'altro l'ambiente viene percepito e manipolato da parte degli agenti considerando le limitazioni imposte dall'artefatto stesso.

Al di là degli artefatti, la nozione di workspace gioca un ruolo importante in A&A. Un workspace è un contenitore logico di agenti e artefatti, e può essere utilizzato per strutturare tutti gli insiemi di entità, definendo una topologia dell'ambiente di lavoro, e fornendo un modo per definire e regolare le interazioni all'interno di questo.

Come detto, quindi, gli agenti utilizzano gli artefatti per interagire tra loro e per svolgere le loro attività. Per capire come avviene questa interazione, analizziamo le caratteristiche di questi due tipi di entità.

2.1 Nozione di Agente

Gli agenti sono entità computazionali autonome che incapsulano il flusso di controllo (insieme al criterio che lo governa), non possiedono un'interfaccia e non possono essere né controllati né invocati.

Essendo un'entità pro-attiva (che per definizione 'agisce per fare accadere qualcosa', invece che aspettare che accada qualcosa), un agente esegue un qualche tipo di attività di lavoro, a livello individuale o a livello collettivo, tipicamente con l'obiettivo di soddisfare un qualche goal (oppure eseguire un qualche compito o task) individuale o sociale (dalla caratteristica di autonomia di un agente derivano infatti anche le caratteristiche di 'socialità' e interattività). Gli agenti in realtà non hanno necessariamente bisogno di avere un goal per poter essere considerati come tali. Essi governano il processo computazionale del sistema multi-agente, portandone avanti il comportamento in maniera distribuita, secondo un certo criterio: questo criterio non deve essere per forza un goal o un task. I sistemi multi-agente costituiti da agenti che hanno un goal in realtà sono solamente una delle possibilità che si hanno per definire il criterio di funzionamento di un agente.

Le attività eseguite dagli agenti, da un punto di vista astratto, sono da considerare come l'esecuzione di una sequenza di azioni, che in base al modello A&A possono essere approssimativamente classificate come:

- internal actions (azioni interne);
- communicative actions (azioni di comunicazione), riguardanti comunicazioni dirette con uno o più agenti attraverso un qualche tipo di ACL

(linguaggio di comunicazione per agenti); questo tipo di azioni può comportare un cambio dello stato degli agenti coinvolti nella comunicazione;

- pragmatic actions (azioni pragmatiche), cioè azioni svolte all'interno dei working environments e riguardanti la costruzione, la condivisione e l'utilizzo di artefatti, dei quali viene modificato lo stato.

Nonostante la sua specifica architettura cognitiva, per poter eseguire azioni sugli artefatti e percepirne gli eventi osservabili, un agente deve essere 'situato' in un working environment: ogni agente è strettamente collegato con l'ambiente in cui vive e in cui interagisce. Per questo motivo è stata introdotta la nozione di *agent body* ('corpo dell'agente'): questo funziona da tramite attraverso il quale la mente dell'agente (cioè quella parte progettata e programmata in base a un determinato tipo di modello/architettura cognitiva) può da un lato percepire gli stimoli provenienti dal working environment, dall'altro condizionare e modificare l'ambiente stesso compiendo delle azioni. Questo tipo di nozione è essenziale per separare (per scopi di progettazione) la mente dell'agente dal working environment in cui l'agente è situato, in modo da essere in grado di utilizzare A&A con diversi tipi di modelli di programmazione per la mente dell'agente.

Il corpo di un agente contiene attuatori per eseguire azioni sul working environment, e un insieme dinamico di sensori per percepire gli stimoli provenienti sempre dall'ambiente. In particolare, i sensori giocano un ruolo fondamentale, cioè quello di 'memoria percettiva', tenendo traccia degli stimoli recepiti, eventualmente applicando filtri e particolari tipi di politiche di 'buffering'. Gli agenti possono essere considerati anche come entità reattive, dal momento che necessitano di avere una rappresentazione il più possibile aggiornata e adeguata del mondo, del quale devono cercare di registrare il maggior numero di cambiamenti possibili.

2.2 Nozione di Artefatto

Un artefatto è un'entità computazionale passiva, dinamica e con stato, progettata per incapsulare e fornire una qualche funzionalità ad altre entità (gli agenti). La funzionalità di un artefatto è strutturata in termini di operazioni, la cui esecuzione può essere innescata da parte degli agenti attraverso una *usage interface* ('interfaccia d'uso') dell'artefatto stesso: questa è composta dall'insieme delle operazioni fornite dall'artefatto. Ogni operazione è identificata da un'etichetta, e da una lista di parametri di ingresso. L'interfaccia d'uso può cambiare dinamicamente in base allo stato dell'artefatto; in altre

parole, è possibile progettare artefatti che mostrano diverse interfacce d'uso in base al loro stato di funzionamento. Oltre all'insieme delle operazioni, l'interfaccia d'uso può presentare anche una serie di proprietà osservabili, cioè proprietà i cui valori dinamici possono essere osservati da parte degli agenti senza dover necessariamente interagire con l'artefatto.

Un'operazione è l'elemento base su cui è strutturata la funzionalità di un artefatto. L'esecuzione dell'operazione può portare sia a cambiamenti nello stato interno (quindi non osservabile) dell'artefatto, sia alla generazione di un insieme di eventi osservabili che possono essere percepiti dagli agenti che stanno utilizzando o anche semplicemente osservando l'artefatto. Come già accennato, le proprietà osservabili sono attributi dinamici e persistenti che appartengono a un artefatto e che possono essere osservate dagli agenti senza interagire con l'artefatto stesso. Gli eventi (osservabili) sono informazioni non-persistenti, considerabili come dei segnali che possono anche avere un contenuto informativo.

Un'operazione può essere associata a una 'guardia' (una condizione che deve essere vera per poter eseguire l'operazione stessa, e che riguarda lo stato interno dell'artefatto), e può essere composta da più step che devono essere eseguiti atomicamente per evitare possibili interferenze. Questo approccio rende possibile l'esecuzione di più operazioni in modo concorrente sullo stesso artefatto, mantenendo la mutua esclusione per l'accesso allo stato di quest'ultimo.

Un'ulteriore caratteristica degli artefatti è rappresentata dalla possibilità di collegarli tra loro, attraverso delle *link interfaces* (interfacce di collegamento), analoghe alle interfacce degli artefatti del mondo reale. Il collegamento è supportato anche per artefatti che appartengono a workspace separati, eventualmente residenti su nodi diversi della rete.

Infine, analogamente a quanto accade nella realtà con gli artefatti (quindi attrezzi, macchinari, ecc), in A&A ogni artefatto deve avere in dotazione un manuale.

2.2.1 Il manuale

Il manuale di un artefatto ha lo scopo di descriverne

- la funzione, cioè il fine per cui è stato progettato l'artefatto, quindi le funzionalità che è in grado di offrire a chi lo volesse utilizzare, non necessariamente legate in modo diretto alle operazioni messe a disposizione;
- l'interfaccia d'uso, quindi la 'forma' osservabile dell'artefatto;

- le ‘operating instructions’ (cioè le operazioni, una sorta di protocollo d’uso), che sono la descrizione delle procedure che un agente deve seguire per interagire in modo significativo con un artefatto, cioè per poterne sfruttare correttamente le funzionalità. Se la funzione descrive cosa si può ottenere con un certo artefatto (what), le operating instructions descrivono come sfruttare un artefatto (how).

Partendo proprio da questo ultimo punto, si può affermare che lo stato e il comportamento di un artefatto, insieme agli effetti delle azioni degli agenti sull’ambiente (attraverso l’uso dell’artefatto stesso), possono dipendere dall’ordine d’esecuzione delle varie operazioni. Le operating instructions, oltre a un’informazione sintattica, possono anche racchiudere un qualche tipo di informazione semantica utile per agenti intelligenti, che possono utilizzarle per il loro practical reasoning. Gli artefatti sono concettualmente simili ai dispositivi che vengono utilizzati dall’uomo: le operating instructions rappresentano per un agente quello che un manuale rappresenta per un umano, che ne legge le istruzioni per sapere come utilizzare un certo dispositivo.

Questo tipo di caratteristica è fondamentale per pensare di poter sviluppare **sistemi aperti**, in cui gli agenti non hanno una conoscenza a priori di tutti gli artefatti disponibili nel loro workspace, dal momento che agenti e artefatti possono anche essere realizzati da programmatori diversi, e, più in generale, nuove istanze e nuovi tipi di artefatti possono essere creati e quindi inseriti (o anche rimossi) dinamicamente nel workspace; analogamente, nuovi agenti possono entrare a runtime a far parte di un determinato workspace, del quale non conoscono nulla a priori.

2.3 Nozione di Sistema Multi-Agente

In A&A il Sistema Multi-Agente (MAS) è un sistema computazionale composto da agenti e artefatti. Entrambi sono entità computazionali situate, per cui anche i MAS sono sistemi computazionali situati: un MAS è sempre immerso in un ambiente, e pensato (o progettato, o modellato) separatamente rispetto a questo ambiente.

Essendo costituito da agenti e artefatti, e considerando le caratteristiche di queste entità, nel complesso risulta che un MAS possiede un comportamento che è il risultato dell’interazione tra entità autonome e indipendenti (agenti) ed entità reattive e funzionali (artefatti).

All’interno di un MAS sono ammessi quattro diversi tipi di interazione: comunicazione (agenti che comunicano con agenti), operazione (agenti che utilizzano artefatti), composizione (artefatti che si collegano ad altri artefatti), presentazione (artefatti che si ‘manifestano’ agli agenti). Il MAS può in-

teragire con l'ambiente esterno, e queste interazioni possono essere attribuite o ai singoli agenti e artefatti, oppure all'intero MAS, a seconda del livello di astrazione desiderato.

2.4 Agenti pianificatori

Il manuale di un artefatto può essere di fondamentale importanza per un agente intelligente che si trovi in un sistema aperto, e che abbia la necessità di soddisfare un certo goal. La pianificazione, per un agente A&A, può rappresentare un aiuto per sfruttare al meglio le informazioni contenute nei manuali degli artefatti presenti nell'ambiente in cui si trova.

Prima di affrontare questo argomento e vedere in che modo il planning possa essere sfruttato, nella prossima sezione verrà presentata una panoramica generale sulla pianificazione, con la descrizione dei suoi concetti base e delle principali tipologie esistenti.

3 Il planning

Un problema di pianificazione consiste nel determinare un piano, ossia un insieme (parzialmente o totalmente) ordinato di azioni necessarie a raggiungere lo stato 'desiderato', a partire da uno stato iniziale, un insieme di azioni eseguibili e un obiettivo da raggiungere (goal).

Nel planning classico (tradizionale) si considerano solo ambienti completamente osservabili, deterministici, finiti e statici. Il planning non-classico tratta invece ambienti parzialmente osservabili o stocastici, e coinvolge un insieme diverso di agenti e di algoritmi.

La pianificazione può essere considerata come un processo per il calcolo dei diversi passi di una procedura di soluzione di un problema di pianificazione, che è non decomponibile (ci può essere interazione tra i sottoproblemi) e reversibile (le scelte fatte durante la generazione del piano sono revocabili tramite backtracking). Un pianificatore è completo quando riesce sempre a trovare la soluzione se esiste, ed è corretto quando la soluzione trovata porta dallo stato iniziale al goal finale in modo consistente con eventuali vincoli.

Il processo di applicazione della procedura di soluzione è irreversibile (l'esecuzione delle azioni determina spesso un cambiamento di stato non reversibile) e non deterministico (il piano può avere un effetto diverso quando applicato al mondo reale, che è spesso imprevedibile; in questo caso è possibile rifare il piano solo parzialmente, oppure invalidarlo tutto a seconda del problema).

3.1 La pianificazione automatica: concetti base

Un pianificatore automatico è un agente intelligente che opera in un certo dominio e che date una rappresentazione dello stato iniziale e del goal, e una descrizione formale delle azioni eseguibili, sintetizza dinamicamente il piano di azioni necessario per raggiungere il goal.

[Rappresentazione dello stato] Occorre fornire al pianificatore un modello del sistema su cui opera. In genere lo stato è rappresentato in forma dichiarativa con una congiunzione di formule atomiche che esprime la situazione di partenza. Lo stato di un sistema spesso può essere osservato solo in modo parziale per una serie di motivi:

- alcuni aspetti possono essere non osservabili;
- il dominio può essere troppo vasto per essere rappresentato nella sua interezza (limitate capacità computazionali per la rappresentazione);
- le osservazioni possono essere soggette a rumore e quindi si hanno delle osservazioni parziali o imperfette;
- il dominio può essere troppo dinamico per consentire un aggiornamento continuo della rappresentazione.

[Rappresentazione del goal] Per rappresentare il goal si utilizza lo stesso linguaggio formale con cui si esprime lo stato iniziale. In questo caso la congiunzione rappresenta una descrizione parziale dello stato finale che si vuole raggiungere: descrive solo le condizioni che devono essere verificate affinché il goal sia soddisfatto.

[Rappresentazione delle azioni] È necessario fornire al pianificatore una descrizione formale delle azioni eseguibili sul sistema detta ‘Teoria del Dominio’. Ciascuna azione è identificata da un nome e modellata in forma dichiarativa per mezzo di precondizioni e postcondizioni. Le precondizioni rappresentano le condizioni che devono essere verificate affinché l’azione possa essere eseguita; le postcondizioni rappresentano gli effetti dell’azione stessa sul mondo. Spesso la Teoria del Dominio è costituita da operatori con variabili che definiscono classi di azioni. A diverse istanziazioni delle variabili corrispondono diverse azioni.

3.2 La pianificazione classica

La pianificazione classica è un tipo di pianificazione off-line che produce l'intero piano prima di eseguirlo, lavorando su una rappresentazione istantanea (snapshot) dello stato corrente. È basata su alcune assunzioni forti:

- tempo atomico di esecuzione delle azioni;
- determinismo degli effetti;
- stato iniziale completamente noto a priori;
- esecuzione del piano unica causa di cambiamento del mondo.

3.2.1 Planning Deduttivo

La tecnica di pianificazione deduttiva genera il piano come dimostrazione di un teorema. Il difetto principale è che la rappresentazione della conoscenza e il mantenimento della verità delle asserzioni sono difficili e computazionalmente poco efficienti. Un algoritmo che si basa sul planning deduttivo è il Situation Calculus: il cosiddetto *Frame Problem* è un problema che consiste nel dover specificare esplicitamente tutte le proprietà che cambiano nel mondo dopo una transizione di stato, e anche quelle che non cambiano; al crescere della complessità del dominio, il problema della rappresentazione della conoscenza diventa intrattabile.

3.2.2 Planning mediante ricerca

La pianificazione classica non deduttiva utilizza linguaggi specializzati per rappresentare stati, goal e azioni, e gestisce la generazione del piano come un problema di ricerca. La ricerca può essere effettuata:

- nello spazio degli stati o situazioni: nell'albero di ricerca ogni nodo rappresenta uno stato e ogni arco un'azione;
- nello spazio dei piani: nell'albero di ricerca ogni nodo rappresenta un piano parziale e ogni arco un'operazione di raffinamento del piano.

Ricerca nello spazio degli stati - STRIPS

La ricerca nello spazio degli stati è il cosiddetto planning lineare. Un pianificatore lineare riformula il problema di pianificazione come problema di ricerca nello spazio degli stati e utilizza le strategie di ricerca classiche.

L'algoritmo di ricerca può essere

- forward: se la ricerca avviene in modo progressivo partendo dallo stato iniziale fino al raggiungimento di uno stato che soddisfa il goal;
- backward: quando la ricerca è attuata in modo regressivo a partire dal goal, fino a ridurre quest'ultimo in sottogoal soddisfatti dallo stato iniziale.

Il problema principale di quest'ultimo algoritmo di ricerca è quello di definire come si effettua la riduzione di un goal in sottogoal (*goal regression*) tenendo conto della forma delle regole. Considerando le definizioni di azione rilevante (permette di raggiungere uno dei congiunti del goal) e azione consistente (non invalida uno dei congiunti del goal), si può affermare che la regressione di un dato goal G attraverso un'azione A consistente e rilevante porta al seguente risultato: ogni effetto positivo di A che compare in G viene cancellato, e ogni preconditione per A è aggiunta al goal (se non è già presente). L'algoritmo di ricerca backward quindi, data la lista G di tutti i goal e sottogoal del problema e l'insieme delle regole R , applica la regressione per ogni regola R_j per cui esiste un goal G_i che appartiene agli effetti positivi di R_j , e questo fino a raggiungere un sottogoal soddisfatto dallo stato iniziale.

STRIPS. È considerato come l'antenato degli attuali sistemi di pianificazione. È un linguaggio per la rappresentazione di azioni, la cui sintassi è molto più semplice del Situation Calculus. In STRIPS lo stato viene rappresentato mediante una congiunzione di predicati in un linguaggio del prim'ordine; per la rappresentazione del goal vale lo stesso discorso, con in più la possibilità di utilizzare delle variabili; infine per rappresentare le azioni, in aggiunta a una lista di parametri, vengono utilizzate 3 liste distinte: una di precondizioni (predicati che devono essere veri per applicare l'azione), una *delete list* (predicati che diventano falsi come risultato dell'azione) e una *add list* (predicati che diventano veri come risultato dell'azione). Add e delete list possono anche essere rappresentate come *Effect list* con predicati positivi o negativi. Con la cosiddetta *STRIPS assumption* viene risolto il Frame Problem: tutto ciò che non è specificato nelle liste degli effetti resta immutato.

L'algoritmo STRIPS utilizza un planner basato su ricerca backward. Si assume che lo stato iniziale sia completamente noto (*Closed World Assumption*: tutto ciò che non è dichiarato nella rappresentazione dello stato iniziale è falso), e si utilizzano due strutture dati: lo stack dei goal, e la descrizione dello stato corrente. A ogni passo del processo di soluzione si cerca di risolvere il goal in cima alla pila: quando si ottiene una sequenza di operatori che lo soddisfa la si applica alla descrizione corrente dello stato ottenendo una

nuova descrizione. Si cerca poi di soddisfare l'obiettivo che è in cima alla pila partendo dalla situazione prodotta dal soddisfacimento del primo obiettivo. Il procedimento continua fino allo svuotamento della pila.

Ricerca nello spazio dei piani - Partial Order Planning

La ricerca nello spazio dei piani viene effettuata dai pianificatori non lineari; questi sono algoritmi di ricerca che gestiscono la generazione di un piano come un problema di ricerca nello spazio dei piani e non più degli stati. L'algoritmo non genera più il piano come una successione lineare (completamente ordinata) di azioni per raggiungere i vari obiettivi, ma nell'albero di ricerca ogni nodo rappresenta un piano parziale e ogni arco un'operazione di raffinamento del piano. Un pianificatore non lineare generativo assume che lo stato iniziale sia completamente noto. Generalmente viene adottata la tecnica del Least Commitment sugli ordinamenti: si impongono sempre e solo i vincoli che sono strettamente necessari, cosa che evita molti backtracking.

Un piano non lineare è rappresentato come

- un insieme di azioni (istanze di operatori),
- un insieme (non completo) di ordinamenti fra le varie azioni, e
- un insieme di 'causal link' (servono a memorizzare il motivo per cui un certo operatore è stato introdotto nel piano, così da affrontare in modo efficiente il problema dell'interazione tra goal).

Il piano iniziale è composto da due azioni fittizie, *start* (senza precondizioni, e come effetti una descrizione completa dello stato iniziale) e *stop* (con precondizione il goal del planner), con la relazione $start < stop$. Ad ogni passo si incrementano l'insieme degli operatori, degli ordinamenti parziali (non vengono imposti ordinamenti non richiesti) e dei causal link fino a che tutti i goal sono risolti. Una soluzione è un insieme di operatori parzialmente specificato e parzialmente ordinato. Per ottenere un piano effettivo si converte l'ordine parziale in uno tra i diversi ordini totali possibili (operazione di linearizzazione).

L'algoritmo di ricerca nello spazio dei piani maggiormente utilizzato è il Partial Order Planning.

3.2.3 Planning Gerarchico

I pianificatori gerarchici sono algoritmi di ricerca che gestiscono la creazione di piani complessi a diversi livelli di astrazione, considerando i dettagli più

semplici solo dopo aver trovato una soluzione per i più difficili. È possibile definire dei valori di criticità alle precondizioni delle varie operazioni, e definire sia operatori atomici che macro operatori. Tutti gli operatori sono definiti ancora con precondizioni ed effetti.

Tra gli algoritmi di pianificazione gerarchica più diffusi, ci sono STRIPS-like e Partial Order. La pianificazione consiste nel generare un piano che porta da uno stato molto vicino allo stato iniziale ad uno stato molto vicino allo stato finale. Questo piano viene poi rifinito con una ricerca di più basso livello che tiene conto dei dettagli tralasciati fino a quel momento. Quindi un algoritmo gerarchico deve essere in grado di pianificare a livello alto, ed espandere piani astratti in piani concreti.

Il raffinamento dei piani è un aspetto particolarmente importante: questo processo avviene applicando una *decomposizione delle azioni*, che riduce un'azione di alto livello a un insieme ordinato di azioni di livello più basso. Quindi la decomposizione delle azioni rappresenta la conoscenza su come rappresentare le azioni stesse. Questo processo continua fino a che nel piano rimangono solamente azioni primitive, che cioè possono essere eseguite automaticamente dall'agente.

AB-STRIPS. È un pianificatore gerarchico che usa la definizione delle azioni di STRIPS dove in più a ciascuna precondizione è associato un valore di criticità che consiste nella sua difficoltà di raggiungimento. La pianificazione procede a livelli in una gerarchia di spazi di astrazione in ciascuno dei quali vengono ignorate le precondizioni dei livelli di difficoltà inferiore. AB-STRIPS esplora interamente lo spazio di un determinato livello di astrazione prima di passare ad un livello più dettagliato; viene implementata in questo modo una ricerca in lunghezza: ad ogni livello di astrazione viene generato un piano completo.

3.3 Pianificazione e azione nel mondo reale

La pianificazione in ambienti non-deterministici e dinamici impone di considerare che gli effetti delle azioni possano non essere quelli previsti e che il mondo sia cambiato anche per cause non dovute alle azioni dell'agente. Si parla di quattro metodi di pianificazione:

- pianificazione senza sensori: l'algoritmo dovrà assicurarsi che il piano raggiunga l'obiettivo in tutte le possibili circostanze;
- pianificazione condizionale: si costruisce un piano condizionale con più rami per le diverse contingenze che potrebbero verificarsi;

- monitoraggio di esecuzione e ripianificazione: periodica verifica dello stato del mondo ed eventuale ripianificazione;
- pianificazione continua: costante verifica dello stato del mondo, generazione di piani ed esecuzione.

3.3.1 La pianificazione condizionale

Un pianificatore condizionale è un algoritmo di ricerca che genera diversi piani alternativi per ciascuna fonte di incertezza del piano. Un piano condizionale è quindi costituito da: azioni causali, azioni di sensing per le verifiche, e diversi piani parziali alternativi di cui uno solo verrà eseguito a seconda dei risultati delle verifiche. In un pianificatore condizionale si può avere un'esplosione computazionale dell'albero di ricerca nel caso di problemi con un numero di contesti alternativi elevato; inoltre un piano completo che tenga conto di ogni possibile contingenza potrebbe richiedere molta memoria. Un altro problema è che non sempre è possibile conoscere a priori tutti i contesti alternativi, per cui spesso si combina l'approccio condizionale con l'approccio probabilistico dove si assegnano dei valori di probabilità alle varie alternative e si pianifica solo per quelle più probabili.

3.3.2 La pianificazione reattiva

I pianificatori reattivi sono algoritmi di pianificazione on-line, capaci di interagire con il sistema in modo da affrontare il problema della dinamicità e del non determinismo dell'ambiente. I pianificatori reattivi discendono dai sistemi reattivi 'puri' che evitano del tutto la pianificazione ed utilizzano semplicemente la situazione osservabile come uno stimolo per reagire. Questi pianificatori osservano il mondo in fase di pianificazione per l'acquisizione di informazione non nota, monitorano l'esecuzione delle azioni e ne verificano gli effetti, e spesso alternano il processo di pianificazione a quello di esecuzione reagendo ai cambiamenti di stato.

4 Planning e A&A

Nella sezione sul meta-modello A&A si è accennato al discorso di quanto possa essere importante il manuale di un artefatto per un agente intelligente. Alcuni principi e concetti del planning tradizionale possono essere utilizzati per permettere a un agente A&A di sfruttare al meglio il manuale di un artefatto.

4.1 Dal planning al manuale

Si consideri un sistema aperto, in cui agenti e artefatti possono essere realizzati da programmatori diversi. Gli agenti possono entrare o uscire a runtime nel working environment, così come gli artefatti possono essere creati, distrutti, o modificati dalle azioni degli agenti. In uno scenario come questo, caratterizzato da dinamicità e non determinismo, il manuale rappresenta uno strumento di fondamentale importanza per gli agenti, che a priori non conoscono nulla riguardo agli artefatti, e che, nel caso in cui abbiano un determinato goal da soddisfare, possono consultare il manuale di tutti gli artefatti presenti nel workspace per conoscerne le funzioni e le operazioni, cercando di capire se e in che modo è possibile sfruttarli per raggiungere il loro obiettivo.

Si consideri ora un agente (con un certo goal da soddisfare) che entra in un workspace dove ci sono alcuni artefatti, dei quali è possibile consultare il manuale. L'agente, avendo come obiettivo quello di soddisfare un certo goal, potrebbe prima cercare informazioni tra le proprietà osservabili degli artefatti, pensando che il nome del goal potrebbe coincidere con il nome di una di queste; se non dovesse avere successo, l'agente potrebbe cercare una qualche operazione di un artefatto tale che uno dei suoi effetti (o anche il nome dell'operazione stessa) coincida con il goal da soddisfare (si trascuri al momento la questione di come il goal dell'agente possa corrispondere con l'effetto di un'operazione: il problema sulle ontologie verrà discusso successivamente); infine, come ultima possibilità, un agente potrebbe avere la capacità di memorizzare gli artefatti utilizzati (insieme alle loro funzionalità/operazioni), per poter essere in grado, in futuro, di creare istanze di questi artefatti nel caso in cui non esistessero al momento nel workspace, ma fossero utili per soddisfare il suo goal.

Nel momento in cui un agente decide di utilizzare una certa operazione di un artefatto, può capitare che una delle precondizioni di questa operazione non sia soddisfatta dallo stato attuale. A questo punto, come nel caso di un planner (agente che effettua planning), la condizione non soddisfatta diventa il nuovo goal dell'agente A&A, che cerca quindi di trovare nuove operazioni

(non necessariamente dello stesso artefatto) che possano permettergli di soddisfare il suo attuale goal, e quindi alla fine anche il goal di partenza, proprio come visto nella precedente sezione in corrispondenza della descrizione del planning mediante ricerca nello spazio degli stati (algoritmo di ricerca backward).

Il planning tradizionale può rappresentare uno spunto per la definizione della sintassi attraverso la quale esprimere le operazioni di un artefatto. Riprendendo i concetti base della pianificazione automatica, è possibile creare un parallelo con il caso in esame; si può immaginare infatti che un planner sia paragonabile ad un agente del meta-modello A&A:

- entrambi hanno almeno un goal da soddisfare;
- nel caso del planner lo stato iniziale è rappresentato dall'intera conoscenza del mondo, mentre nel caso dell'agente A&A è costituito dalla sua base di conoscenza (insieme, potenzialmente, allo stato osservabile di tutti gli artefatti presenti nel working environment dell'agente);
- il planner per soddisfare il suo goal deve applicare una o più azioni (da scegliere tra tutte quelle applicabili), così come l'agente deve sfruttare gli artefatti, e in particolare le loro operazioni; le azioni del planning possono essere quindi paragonate alle operazioni di un artefatto:
 - così come le azioni del planning sono rappresentate tramite precondizioni e postcondizioni (o effetti), ha senso esprimere anche le operazioni di un artefatto in questo modo;
 - nel caso del planning le precondizioni e le postcondizioni rappresentano rispettivamente le condizioni che devono essere vere prima e che diventano vere dopo l'applicazione dell'azione; in questo caso precondizioni ed effetti possono riguardare l'intero mondo (o ambiente) in cui agisce il planner. Nel caso di A&A c'è un'importante differenza: le condizioni che vengono espresse riguardano lo stato dell'artefatto cui fanno riferimento; in particolare le precondizioni sono delle condizioni che riguardano lo *stato osservabile* dell'artefatto, mentre gli effetti ne definiscono il cosiddetto ' Δ stato_osservabile', rappresentato da eventuali variazioni di proprietà osservabili o generazione di eventi; in pratica si usa lo stesso principio della 'STRIPS assumption', che prevede che tutto ciò che non viene specificato negli effetti resta immutato, cioè nella stessa situazione in cui si trovava prima dell'utilizzo dell'operazione;

- inoltre si ritiene opportuno dover specificare, per ogni operazione, una lista di parametri di ingresso necessari per la sua esecuzione.

Da questa corrispondenza è emerso un aspetto particolarmente importante, cioè il diverso ‘ambito’ di definizione delle azioni (nel planning AI) e delle corrispondenti operazioni (per gli artefatti A&A): nel primo caso si tratta dell’intero mondo, nel secondo caso invece si tratta di una parte limitata dell’ambiente, cioè del singolo artefatto che mette a disposizione una determinata operazione. Questo concetto è molto importante: nell’AI tradizionale si tratta di far transitare tutto il sistema in un certo stato, che viene considerato come un unico blocco, mentre questo approccio permette di separare il mondo in tante piccole parti (ogni singolo artefatto); questo può permettere di decomporre il ragionamento in sotto-parti, evitando di affrontare sempre la complessità del mondo nella sua totalità.

Quindi, ricapitolando, l’agente A&A rappresenta il pianificatore, e le azioni che ha a disposizione per generare il piano e raggiungere il suo goal sono rappresentate dalle operazioni dei diversi artefatti che ha a disposizione. In particolare, nella notazione del manuale, le precondizioni e gli effetti delle varie funzioni / operazioni sono condizioni che riguardano lo **stato osservabile dell’artefatto** cui si riferiscono; nel caso del planning tradizionale, riguardano invece lo stato dell’intero ambiente (mondo). In questo modo, frammentando il mondo in più parti e non considerandolo come un blocco monolitico, ragionando solo sulle sezioni di interesse, si tenta di ridurre la complessità generale, che in questo modo non viene affrontata nella sua interezza.

4.2 Ontologie

In sistemi totalmente ‘aperti’ le ontologie utilizzate per descrivere da un lato i goal di un agente e dall’altro le funzionalità e le operazioni di un artefatto non coincidono necessariamente, per cui occorre avere forme di ‘interpretazione’, a partire dall’attribuzione di una semantica esplicita sia per i goal, sia per le funzionalità. In questo contesto non verrà affrontato questo problema e quindi non sarà considerata questa fase di interpretazione.

5 Modello del manuale

Nella sezione precedente è stato presentato un possibile modo per rappresentare le operazioni di un artefatto, cioè tramite parametri di ingresso, precondizioni ed effetti; basandosi sulle considerazioni fatte fino a questo momento, viene definito ora un possibile modello per la rappresentazione del manuale di un artefatto. Verrà considerata una versione semplificata del manuale, che comprende solamente le operating instructions (trascurando le funzionalità). Fino a questo momento infatti si è sempre parlato quasi esclusivamente di operazioni, e non di funzionalità di un artefatto: come già accennato anche nella sezione su A&A, le funzionalità offerte da un artefatto non necessariamente coincidono con l'elenco delle operazioni, o meglio, dei controlli presenti nell'interfaccia d'uso, ma sono più astratte e più generali. Nel manuale sarebbe plausibile esprimere anche le funzionalità, ma per il momento si è deciso di limitarne il contenuto al solo protocollo d'uso, quindi le varie operazioni, lasciando all'agente la libertà (ma anche il peso) di combinare tra loro queste ultime - anche di artefatti diversi - per soddisfare il suo goal.

5.1 Grammatica del manuale

Dopo aver discusso sul contenuto semantico del manuale, cioè quali informazioni debbano essere presenti per renderlo utile al practical reasoning di un agente intelligente, in questa sezione viene presentata la sintassi attraverso la quale viene definito il linguaggio per la rappresentazione del manuale. A questo scopo è stata elaborata una grammatica (strumento formale per descrivere in maniera precisa e non ambigua il modo in cui i dati e i programmi vengono rappresentati); i quattro elementi che la caratterizzano sono:

- simboli non terminali

```
<Manual>, <manualBlock>, <obsPropList>, <operation>,  
<operationBlock>, <parameters>, <parametersList>,  
<singleParameter>, <preconditions>, <preconditionsList>,  
<singlePrecondition>, <effects>, <effectsList>,  
<singleEffect>, <singleEffectBlock>, <assignment>, <event>
```

- simboli terminali

- <NUM>: token che definisce tutti i numeri naturali positivi, che possono comparire o nelle precondizioni, o negli effetti (aggiornamento del valore di una proprietà osservabile);

- <NAME>: token usato per definire i nomi dell’artefatto, delle operazioni, delle proprietà osservabili e dei parametri;
- <TYPE>: definizione dei tipi di parametri di ingresso delle operazioni;
- <BOOL_OP>: operatori booleani, che si possono trovare nelle precondizioni;
- <OP>: operatori aritmetici, che si possono trovare nelle espressioni, e in particolare nei casi di aggiornamento del valore di una variabile (effetti);
- <EQUAL>: simbolo ‘=’;
- <AND>: simbolo ‘&’;
- <OR>: simbolo ‘|’;
- <LEFT_ROUND_BRACKET>, <RIGHT_ROUND_BRACKET>, <LEFT_BRACES>, <RIGHT_BRACES>: parentesi tonde e graffe;
- <COLON>, <SEMICOLON>, <COMMA>, <DOT>: simboli di punteggiatura (‘due punti’, ‘punto e virgola’, ‘virgola’, ‘punto’);

- regole di produzione

```

<Manual> ::= 'manual' <NAME>
           <LEFT_BRACES> <manualBlock> <RIGHT_BRACES>
<manualBlock> ::= 'obsProp' <COLON>
                 <LEFT_BRACES> <obsPropList> <RIGHT_BRACES>
                 <SEMICOLON> ( <operation> )+
<obsPropList> ::= <NAME> ( <COMMA> <NAME> )*
<operation> ::= <NAME> <LEFT_BRACES> <operationBlock>
               <RIGHT_BRACES>
<operationBlock> ::= <parameters> <SEMICOLON>
                   <preconditions> <SEMICOLON>
                   <effects> <SEMICOLON>
<parameters> ::= 'params' <COLON> <LEFT_BRACES>
                <parametersList> <RIGHT_BRACES>
<parametersList> ::= <singleParameter>
                    ( <COMMA> <singleParameter> )*
<singleParameter> ::= <NAME> <COLON> <TYPE>
<preconditions> ::= 'pre' <COLON>
                  <LEFT_BRACES>
                  <preconditionsList>
                  <RIGHT_BRACES>

```

```

<preconditionsList> ::= <singlePrecondition>
                        ( <AND> <singlePrecondition> )*
<singlePrecondition> ::= <NAME> <BOOL_OP> (<NAME> | <NUM>)
<effects> ::= 'effects' <COLON>
                <LEFT_BRACES> <effectsList> <RIGHT_BRACES>
<effectsList> ::= <singleEffect>
                ( (<SEMICOLON> | <DOT>)
                  <singleEffect> )*
<singleEffect> ::= <singleEffectBlock>
                |
                <LEFT_ROUND_BRACKET> <singleEffectBlock>
                ( <OR> <singleEffectBlock> )+
                <RIGHT_ROUND_BRACKET>
<singleEffectBlock> ::= <assignment> | <event>
<assignment> ::= <NAME> <EQUAL> <expr>
<event> ::= <NAME> [ <LEFT_ROUND_BRACKET> (<NAME> | <expr>())
                ( <COMMA> (<NAME> | <expr>()) )*
                <RIGHT_ROUND_BRACKET> ]
<expr> ::= (<NAME> | <NUM>) [ <OP> (<NAME> | <NUM>)]

```

- scope (simbolo iniziale)

<Manual>

Questa grammatica descrive un linguaggio di questo tipo:

```

manual ARTIFACT_NAME {
  obsProp: { OBS_PROP_1, ..., OBS_PROP_N };
  operation OPERATION_NAME_1 {
    params: { PARAM_1: TYPE, ..., PARAM_2: TYPE };
    pre: { PRECONDITION_1 & ... & PRECONDITION_N };
    effects: { EFFECT_1; EFFECT_2. EFFECT_3. ... ;EFFECT_N };
  }
  operation OPERATION_NAME_2 {
    ...
  }
}

```

È possibile fare una serie di considerazioni riguardanti la grammatica, e in particolare le regole di produzione:

- come prima cosa nel manuale vengono elencati i nomi delle proprietà osservabili dell'artefatto, che verranno eventualmente utilizzate in seguito nelle definizioni delle varie operazioni, e che in qualche modo potrebbero essere utili all'agente;
 - successivamente vengono definite tutte le operazioni messe a disposizione dall'artefatto: ognuna è rappresentata da 3 strutture, cioè una lista dei parametri di ingresso, una serie di precondizioni e una di effetti:
 - ogni operazione può richiedere uno o più parametri di ingresso, cioè tutti i dati di cui l'agente deve essere in possesso per poterla sfruttare; questi sono rappresentati da una lista di elementi separati da virgola, dove ognuno definisce il nome e il tipo del parametro;
 - le precondizioni sono una congiunzione di espressioni booleane, in cui vengono espresse condizioni riguardanti le proprietà osservabili dell'artefatto, o eventualmente i parametri di ingresso;
 - gli effetti necessitano di una descrizione più elaborata rispetto alle strutture precedenti, e sono espressi mediante una notazione simile a quella utilizzata per la descrizione dei processi nella programmazione concorrente (una sorta di *Process Algebra* dove gli effetti rappresentano i processi). Nel corso dell'esecuzione di un'operazione, gli effetti seguono un determinato ordine, indipendentemente dal fatto che si tratti dell'aggiornamento di una proprietà osservabile, o della generazione di un evento; per questo motivo è necessario poter esprimere come prima cosa la loro corretta successione temporale, e questo avviene elencando i vari effetti come una sequenza di elementi separati dal simbolo ';' (punto e virgola). Inoltre è importante poter esprimere e descrivere in qualche modo situazioni in cui, in base al verificarsi di una o più condizioni booleane, possano essere generati solo alcuni effetti, a discapito di altri: per questo motivo si utilizza il simbolo '|' (or), per separare tra loro gli effetti mutuamente esclusivi, mentre l'insieme (degli effetti separati dal simbolo '|') viene racchiuso all'interno di parentesi tonde (le condizioni booleane che determinano il verificarsi o meno degli effetti non verranno comunque riportate). In questo modo diventa possibile individuare tutte le sequenze di effetti che si possono verificare utilizzando una certa operazione.
- Un'operazione può essere anche composta da una serie di step, e può essere utile evidenziare questo tipo di separazione: in questi

casi, tutti gli effetti dell'operazione sono sempre elencati in un'unica sequenza, ma l'ultimo effetto di ogni step verrà separato del primo effetto dello step successivo tramite il separatore '.' (punto). È possibile che nel corso di un'operazione (ed eventualmente dei vari step di cui può essere composta) determinati effetti possano essere ripetuti più di una volta: per esempio una proprietà osservabile viene aggiornata diverse volte (con valori diversi), così come un evento può essere generato più volte. La scelta migliore potrebbe essere quella di evitare di ripetere tutte le volte lo stesso effetto, a meno che questo venga aggiornato nel corso di step diversi della stessa operazione; quindi se un effetto venisse aggiornato più volte nel corso di un'unico step, oppure di un'unica operazione (non composta), l'effetto verrebbe inserito nella sequenza solo in corrispondenza dell'ultimo aggiornamento, cioè quello definitivo, perché potrebbe essere effettivamente l'unico evento di interesse per l'agente.

Entrando nello specifico, nel caso di aggiornamento di una proprietà osservabile, viene specificata l'espressione aritmetica che determina il nuovo valore assunto da quest'ultima, anche se questa scelta può comportare delle complicazioni: potrebbe infatti accadere che il nuovo valore assunto da una proprietà osservabile sia ricavato da una lunga serie di calcoli svolti nel corso dell'operazione, e in questo caso risulterebbe molto difficoltoso (se non impossibile) riassumere in un'unica formula il valore finale assunto dalla proprietà. In questi casi si potrebbe semplicemente riportare il solo nome della proprietà osservabile in questione, per indicare che questa, nel corso dell'operazione, viene in qualche modo modificata. Lo stesso ragionamento vale anche per gli eventi, che sono definiti da un nome, che può essere accompagnato da una serie di parametri, tra i quali solitamente anche il valore (o i valori) di ritorno, che tipicamente è il risultato dell'operazione eseguita.

5.2 Esempi

Come esempio di realizzazione di un manuale, viene considerato il caso di un artefatto **CALCULATOR**: le operazioni messe a disposizione dalla calcolatrice sono quelle di moltiplicazione e divisione, e, in assenza di una proprietà osservabile che svolga il ruolo del display, l'unico effetto di queste operazioni (che richiedono due parametri di ingresso) è semplicemente la generazione di un evento che includa, tra i suoi parametri, anche il risultato.

```

manual CALCULATOR {
  obsProp: {};
  operation MUL {
    params: { P1: double, P2: double };
    pre: {} ;
    effects: { MUL(P1,P2,P1*P2) } ;
  }
  operation DIV {
    params: { P1: double, P2: double };
    pre: { P2 != 0 } ;
    effects: { DIV(P1,P2,P1/P2) } ;
  }
}

```

Consideriamo anche un altro caso, utilizzando lo stesso artefatto, ma con in aggiunta la proprietà osservabile `DISPLAY`, che visualizza il risultato finale delle operazioni. Per complicare un po' la situazione, si può pensare che l'effetto delle varie operazioni sia l'aggiornamento della proprietà osservabile nel caso in cui il risultato sia maggiore di zero, mentre sia la generazione di un evento in caso contrario.

```

manual CALCULATOR {
  obsProp: { DISPLAY };
  operation MUL {
    params: { P1: double, P2: double };
    pre: { DISPLAY == 0 } ;
    effects: { (DISPLAY = P1 * P2 | MUL(P1,P2,P1*P2)) } ;
  }
  operation DIV {
    params: { P1: double, P2: double };
    pre: { P2 != 0 & DISPLAY == 0 } ;
    effects: { (DISPLAY = P1 / P2 | DIV(P1,P2,P1/P2)) } ;
  }
}

```

6 Conclusioni e sviluppi futuri

Lo scopo di questo approfondimento è stato quello di presentare un semplice modello per la definizione del manuale di un artefatto in un ambiente A&A. Il modello è stato presentato tramite la grammatica formale che ne definisce il linguaggio. Questo modello potrebbe essere migliorato ulteriormente, andando a definire strutture più elaborate soprattutto per la parte riguardante l'espressione degli effetti; per esempio, nella definizione di un'operazione, si potrebbe introdurre un'ulteriore struttura (in aggiunta a parametri di ingresso, precondizioni, ed effetti) da utilizzare per descrivere meglio le operazioni composte da più step, andando eventualmente a evidenziare le possibili sequenze di step da eseguire per completare l'operazione, compresa la definizione dei causal link tra i vari step (ricollegandosi ad alcuni concetti del Partial Order Planning e del Planning Gerarchico).

Con l'esame 'Progetto di Sistemi Informatici' verrà ripreso il lavoro svolto con questo approfondimento, con l'obiettivo di utilizzare il modello del manuale per la realizzazione di un esempio applicativo concreto, che sfrutterà l'integrazione Jason/Cartago.