**Nome e Cognome:** **Matricola:**

# Corso di Intelligenza Artificiale

Anno Accademico 2009/2010

Esempio di Compitino – Seconda Parte

## Istruzioni

- Scrivere *Nome*, *Cognome* e *Matricola* su **ogni** foglio (solo pagine **dispari**).
- Scrivere la risposta nello spazio bianco al di sotto della domanda; Non è possibile allegare fogli aggiuntivi, quindi cercate di essere chiari e non prolissi.
- In caso di errori indicate chiaramente quale parte della risposta deve essere considerata; annullate le parti non pertinenti.
- Assicurarsi che non manchi alcun foglio al momento della consegna.

# Domande

## domanda 1

Dal punto di vista della rappresentazione della conoscenza e della inferenza nella logica del primo ordine, caratterizzare il linguaggio Prolog.

## domanda 2

Discutere il vantaggio dei pianificatori non-lineari rispetto a quelli lineari, esemplificando, se possibile, i concetti espressi. Descrivere nel maggior dettaglio possibile il pianificatore POP.

*Esempio di risposta:*

Gli algoritmi di planning lineari sono forzati a costruire incrementalmente un piano che corrisponde ad una sequenza ordinata di azioni, anche se ciò non è strettamente necessario. Un pianificatore non-lineare si preoccupa di far emergere solo i vincoli di ordinamento fra le azioni (di un piano) che necessariamente devono essere soddisfatte, restituendo all'utente un ordinamento parziale fra le azioni. Ogni ordinamento totale delle azioni consistente con tale ordine parziale costituisce una soluzione al problema. Un pianificatore lineare assume implicitamente che gli eventuali sottogoal possano essere considerati indipendenti e quindi che un piano possa essere ottenuto concatenando le sottosequenze di azioni che separatamente raggiungono i sottogoal. Tale assunzione è all'origine della Anomalia di Sussman illustrata a lezione tramite il mondo dei blocchi. Questa cosidera un problema di pianificazione con stato iniziale dato da `on(B,Table), on(A,Table), on(C,A), clear(B), clear(C), handempty`, e goal `on(B,C), on(A,B)`. Un pianificatore lineare tenterà di risolvere prima di tutto il sottogoal `on(B,C)` o il sottogoal `on(A,B)`, senza curarsi del fatto che la soluzione per tale sottogoal ( `[unstack(C,A), putdown(C), stack(B,C)]` per il sottogoal `on(B,C)`, e `[unstack(C,A), putdown(C), stack(B,A)]` per il sottogoal `on(B,C)`) debba successivamente essere disfatta in parte per permettere il raggiungimento del secondo sottogoal. POP non incorre in tale rischio, in quanto non assume l'indipendenza dei sottogoal e tramite il principio del minimo impegno introduce solo azioni o vincoli fra azioni che sono strettamente necessari. Una conseguenza del modo di operare dei pianificatori lineari è l'incapacità di trovare un piano nel caso in cui per soddisfare un sottogoal si consumi una risorsa non ripristinabile, ma necessaria anche per portare a termine un altro sottogoal: una volta consumata per soddisfare il primo sottogoal, la risorsa non sarà più disponibile per soddisfare il sottogoal successivo. Anche in questo caso POP riesce a trovare una piano grazie alla gestione delle minacce: l'uso prematuro della risorsa minaccia il raggiungimento del sottogoal e pertanto il suo utilizzo viene posticipato nel tempo fino alla eliminazione della minaccia.

POP raggiunge un piano, se questo esiste, tramite una ricerca nello spazio dei piani. In particolare, un piano è definito da 3 entità principali: 1) un insieme di passi $\{S_1, \ldots, S_n\}$ dove ogni passo è caratterizzato dalla descrizione di un operatore, precondizioni e postcondizioni; 2) un insieme di link causali $\{\ldots, (S_i, c, S_j), \ldots\}$, dove un link causale indica che uno dei propositi del passo $S_i$ è di raggiungere la precondizione $c$ del passo $S_j$; 3) un insieme di vincoli di ordinamento $\{\ldots, S_i < S_j, \ldots\}$, nel caso in cui il passo $S_i$ debba precedere il passo $S_j$.

Un piano non-lineare è completo sse

- Ogni passo menzionato in 2) e 3) è in 1);
- Se $S_j$ ha prerequisito $c$, allora esiste un link causale in 2) nella forma $(S_i, c, S_j)$ per

qualche $S_i$;

- Se $(S_i, c, S_j)$ è in 2) e il passo $S_k$ è in 1), e $S_k$ minaccia $(S_i, c, S_j)$ (rende $c$ falso), allora 3) contiene $S_k < S_i$ o $S_k > S_j$.

Il piano iniziale (tipicamente non completo) è costituito da due passi speciali: il passo "start" che avrà solo post-condizioni che coincidono con la rappresentazione dello stato iniziale, e il passo "finish" che avrà solo precondizioni corrispondenti alla rappresentazione del goal. Inoltre il passo "start" deve precedere il passo "finish". POP iterativamente raffina il piano iniziale (non completo) tramite l'aggiunta di un nuovo passo al piano per chiudere una o più precondizioni aperte, o tramite l'aggiunta di un nuovo vincolo ai passi già presenti nel piano parziale.

*Di seguito bisogna riportare una descrizione dell'algoritmo POP. Una prima possibilità consiste nel riportare le 4 funzioni in pseudocodice; una seconda possibilità consiste nel dare una descrizione a parole di POP. In questo caso ricordarsi di non dare nulla per scontato.*

## domanda 3

Descrivere gli ingredienti fondamentali dell'Apprendimento Automatico.

## domanda 4

Dare la definizione di indipendenza condizionale, spiegando il suo ruolo all'interno della inferenza probabilistica, ed in particolare delle Reti Bayesiane.

## domanda 5

Spiegare in dettaglio l'algoritmo Rejection Sampling per l'inferenza probabilistica approssimata. Dimostrarne la consistenza.