Istruzioni per il progetto

1. Configurazione.

La directory lib contiene vari moduli e programmi utili. Per usarli come library(…), lib va messa nella directory di lavoro oppure bisogna copiarne il contenuto nella home (per windows, consultare comunque il manuale) come segue:

<home utente>/lib/prolog

In questo secondo caso è visibile ovunque nella home.
Fissata la configurazione, per usare action occorre importare:

```
:- use_module(library(mr)).
:- consult(library(action)).

Se volete fare type checking (consigliabile) aggiungete:

:- use_module(library(is_a)).
:- use_module(library(action_spec)).
:- discontiguous(type(_)).
:- discontiguous(pred(_)).
type T :- action_spec:type(T).
pred P :- action_spec:pred(P).
```

2. Come usare moduli e specifiche.

2.1. Il programma action.pl.

Non è un modulo, bisogna importarlo con consult è specificato nei files action_if.pl e action_spec.pl;

action_if.pl *è l'interfaccia che il vostro progetto deve implementare*

action_spec.pl specifica i tipi e predicati implementati da action usando i tipi e predicati dell'interfaccia; in prticolare:

- **solution(list(action),number):** solution(-Piano,-C) fornisce il piano che risolve il problema e il costo relative
- prima di chiamare solution bisogna caricare la strategia di ricerca; con **default_strategy** caricate astar + pota_chiusi
- **exec.** Calcola il piano e ne mostra l'esecuzione.

2.2. Il *modulo di ricerca* mr.pl.

E' specificato in search_if.pl (interfaccia) e search_spec.pl (predicati implementati); si tratta di un modulo, da importare con use_module.

L'interfaccia di mr.pl è già implementata da action.pl, eccetto l'euristica h(stato, number) che dovete implementare voi.

3. TESTING E DEBUGGING.

Costruire dei casi di test per i predicati più importanti e conservarli in uno o più file test_xxx.pl da poter riutilizzare in fase di testing di modifiche che vi capiterà di fare. Si veda come esempio il file

test_scimmieebanane.pl. I file di test faranno parte del codice presentato per l'esame.

Se trovate errori usate gli strumenti di debugging implementati. L'attivazione della esecuzione passo passo del modulo ricerca mr si attiva dando il comando show (prima dell'esecuzione) e si disattiva con noshow; durante l'esecuzione passo passo si possono dare ulteriori comandi; h mostra i comandi utilizabili; in particolare, con t si può attivare la trace Prolog.

NB. Nella esecuzione passo passo di mr i nodi nc(Stato, Path, Costo) in frontiera vengono visualizzato con una semplice writeln; può convenire **PERSONALIZZARE** la visualizzazione dei nodi nc(Stato, Path, Costo) implementando il predicato mostra_nodo(+Nodo).
Ad esempio in action.pl mostra_nodo è implementato come segue:

```
mostra_nodo(nc(S, Path, Cost)) :-
     maplist(write, ['cost:',Cost,' stato:', S, '\n']),
     reverse([S|Path], SL),
     states_to_actions(SL, Act),
     write('  azioni:'), writeln(Act).
```

Cioè anziché il nodo viene stampata la sequenza di azioni corrispondente, dal momento che nel debugging della ricerca di un piano è più leggibile; volendo si può aggiungere anche la visualizzazione del Path.

3. Cosa fare.

Oltre a implementare l'interfaccia, dovrete adattare il codice di action.pl alle vostre esigenze; in particolare:

- sviluppare una ragionevole interfaccia utente
- adattare exec in modo da mostrate l'esecuzione mostrando stati ed azioni nel modo più adatto; eventualmente fornire un'animazione
- studiare più euristiche e confrontarle su un insieme di istanze del problema, eventualmente generate in modo random; riportare in tabelle o grafici i risultati sperimentali ottenuti
- specificare tipi e predicati da voi sviluppati; per i predicati indicare modi e Spec.
- Scrivere una relazione di 5-10 pagine che descrive il problema, le soluzioni adottate, le euristiche, e contiene in un'appendice le sperimentazioni effettuate.
- Alla relazione va allegato il codice prolog documentato, inclusi i file di test.

4. Tipi di problemi che si adattano a questo tipo di progetto.

Esempio 1. Delivery. Uno o più robot devono, ad es., consegnare la posta in un ufficio, potete pensare ai piani di Via Comelico. I robot possono chiamare l'ascensore, muoversi lungo i corridoi, entrare nelle stanze e lasciare la posta sulle scrivanie. Si tratta di pianificare la consegna.

Varianti: ad esempio complicare l'esempio dell'aspirapolvere.

Esempio 2. Pathfinding tattico. In un videogioco un agente artificiale (NPC) deve raggiungere una destinazione su un terreno accidentato e con eventuali pericoli; per affrontare le varie difficoltà si può avvalere di mezzi (ad es. sci per i tratti nevosi, scudo speciale per i tratti esposti al tiro nemico, ecc.). Il costo di un tratto di cammino dipende dalle difficoltà e dai mezzi di cui l'agente si può dotare. La pianificazione deve trovare i mezzi e il percorso migliori.

Varianti: al posto del terreno ci può essere un ambiente con stanze su uno o più piani; un fantasma attraversa i muri; ecc.

Esempio 3. Ristorante. Sapendo i tempi di cottura e avendo N fuochi, in che ordine cucinare i piatti per evadere nel modo più rapido un insieme di ordinazioni? Si osservi che lo stato è caratterizzato dalle ordinazioni pendenti.

Sono solo alcuni esempi, inventate voi.