Progetto di ingegneria della conoscenza

FABIO FUCCI 698996
VITO DEBELLIS 702335
ROBERTO CASALE 698995

Obiettivo del progetto

- Sviluppo di un motore di inferenza che permetta di effettuare ragionamento deduttivo e abduttivo a partire da una base di conoscenza di clausole di Horn.
- Linguaggio utilizzato: Python

Il metodo load_kb

- Appartiene alla classe Engine
- Permette di caricare in memoria una base di conoscenza, presente in un file di testo.
- Attraverso la funzione parse permette di effettuare il parsing di un espressione scritta nel nostro linguaggio.
  Un espressione può essere:
  - Una clausola che può essere aggiunta alla base di conoscenza.
  - Una query che può essere passata, come argomento, al metodo prove.

La valutazione della query

- Avviene attraverso il metodo prove, utilizzando il backword chaining.
- L'output sarà una sequenza di clausole di risposta, generate per provare l'obiettivo.
- Il metodo prove prende in input tre flag:
  - occurs_check: specifica se effettuare il controllo di occorrenza durante l'unificazione di due espressioni.
  - prove_one: specifica se fermarsi dopo aver trovato una sola sequenza di clausole di risposte per dimostrare la query.
  - abduce: specifica se effettuare diagnosi abduttiva.

Features aggiuntive

- Gli atomi negati sono risolti tramite negation as failure.
  Per indicare un atomo negato lo si scrive con il suffisso “not_”.
- È possibile usare i simboli di funzione per identificare un individuo attraverso altri individui.
- E’ possibile definire i propri predicati e le proprie funzioni built-in (scritti in Python) per una maggiore efficienza nella valutazione.
- Effettuando diagnosi abduttiva provando “false” è possibile effettuare diagnosi basata su consistenza.
- Il metodo how prende in input un atomo e restituisce l’insieme di regole utilizzate per provarlo.

Dominio applicativo

- Il motore è posto all'interno di un ambiente, inventato da noi, per dimostrarne le features.
- Abbiamo un agente che si muove all'interno di questo ambiente per gestire i conflitti presenti all'interno della base di conoscenza. I comandi vengono passati attraverso un'interfaccia grafica.
  - Il percorso migliore tra due punti è trovato tramite l'algoritmo A*.
  - Essendo presenti più piani all'interno dell'ambiente, scale e ascensori vengono considerati come isole.
  - Per scegliere in quale ordine gestire gli assumibili di un conflitto si utilizza l'algoritmo branch and bound.
  - "muovi(posizione)." permette di muovere l'agente in quella posizione
  - "abduce", "occurs check" e "prove one" permettono di attivare o disattivare i flag da passare al metodo prove.
  - "check_conflicts" permette di gestire un conflitto presente
  - Qualunque comando terminante con un puntino viene considerato come una query per il motore di inferenza.
  - "how atom" restituisce le regole usate per provare atom