Sistemas Especialistas

Jomi Fred Hübner

Departamento de Automação e Sistemas
http://www.das.ufsc.br/~jomi/ia

PPGEAS 2014/1

Sistemas Especialistas

- usado para tomada de decisão
- baseado em conhecimento
  - representado por regras IF ... THEN ...
  - determinado por um especialista
  - em um domínio específico

cláusula de **Horn**

Motor (production system)

- um *loop* contendo:
  1. achar uma regra IF THEN aplicável
  2. resolver conflitos (quanto existem várias regras aplicáveis)
  3. executar a ação correspondente
- até nenhuma regra ser aplicável

Forward chaining

1 **function** solve($\mathcal{Q}, \mathcal{K}$): $\{true, false\}$

Input: um conjunto finito de proposições, $\mathcal{Q} = q_1, q_2, ..., q_n$;
uma base de conhecimento $\mathcal{K}$

Output: true ou false conforme uma base de conhecimento
implica em todos os q_i

2 **while** $\exists_{c \in \mathcal{K}}\ c = p_1 \wedge ... \wedge p_n \rightarrow p$ **and** $\forall_i\ soved(p_i)$ **do**

3 $\quad solved(p) \leftarrow true$

4 $\quad$ **if** $\forall_i\ solved(q_i)$ **then**

5 $\quad\quad$ **return** true

6 **return** false

Raciocínio orientado a **dado**:

P → Q

é lido como

assert(P) ▷ assert(Q)

Backward chaining

1 **function** solve($\mathcal{Q}, \mathcal{K}$): $\{true, false\}$

Input: um conjunto finito de proposições, $\mathcal{Q} = q_1, q_2, ..., q_n$;
uma base de conhecimento $\mathcal{K}$

Output: true ou false conforme uma base de conhecimento
implica em todos os q_i

2 **if** n = 0 **then**
3 **return** true
4 **forall** cláusula $c \in \mathcal{K}$ **do**
5 **if** $c = p_1 \wedge ... \wedge p_m \rightarrow q_1$ **and**
solve($\{p_1, ..., p_m, q_2, ..., q_n\}, \mathcal{K}$) **then**
6 **return** true
7 **return** false

Raciocínio orientado a **meta**:

P → Q

é lido como

goal(Q) ▷ goal(P)

Implementação

- Em Prolog
- Em *shell* específica (CLIPS, JESS, …)
  - http://www.jessrules.com
- …

Memória de trabalho

- (person age:27 home:toronto)
- (olderThan firstArg:john secondArg:mary)

Regras

- Condições (parte IF)
  - `-(person age:{<23 and >6})`
  - `(person age:[`*n*`+4] occupation:`*x*`)` // *n* tem que ter valor, *x* ganha valor
- Conseqüência (parte THEN)
  - ADD
  - REMOVE
  - MODIFY

Exemplos

- IF (student name: *x*)
  THEN ADD (person name: *x*)

- IF (person age: *x* name: *n*)
     (birthday who: *n*)
  THEN MODIFY 1 (age [*x*+1])
       REMOVE 2

- (counter value:1)
  (brick name:A size:10 position:heap)
  (brick name:B size:30 position:heap)
  (brick name:C size:20 position:heap)

- IF (brick position: heap name: *n* size: *s*)
     -(brick position:heap size:{> *s*})
     -(brick position:hand)
  THEN MODIFY 1 (position hand)

- IF (brick position:hand)
     (counter value:*i*)
  THEN MODIFY 1 (position *i*)
       MODIFY 2 (value [*i* + 1])

Características

- o projetista determina o conhecimento, não o fluxo de controle (declarativo)
- segue a proposta de conhecimento como determinante no comportamento do agente [Newell]

Vantagens

- disponibilidade
- durabilidade
- custo
- perigo
- experiências multiplas
- não emocional
- separação conhecimento/ controle
- formalização do conhecimento
  - validação
  - detecção de inconsistências

Algumas referências

- BRACHMAN; LEVESQUE. *Knowledge Representation and Reasoning*. Elsevier, 2004.