Horn-Clauses DSL

Guilherme Azevedo Horn (247127)

Kauan Aprigio Estevão (260205)

André Ribeiro do Valle Pereira(244185)

Contexto

- Permitir a definição de fatos e regras lógicas de forma declarativa, com uma sintaxe próxima da linguagem natural da lógica proposicional.

- Permite declarar fatos e regras, usando and, or e implicação =>

- Permite aplicar funções aos fatos

Knowledge Base (KB)

- Banco de conhecimento com todas as regras e fatos

- Formato da regras : (premissas . conclusão)

- Formato dos fatos : (Fato1 Fato2 Fato3 …)

Mudar a KB

- E possível mudar a kb, adicionando novas regras ou novos fatos. Na verdade

- Adicionando fatos : Uma nova kb e criada, copiando as regras e adicionando os novos fatos

- Adicionando Regras : Uma nova kb e criada, copiando os fatos e adicionando as novas regras

Inferência

- Como mencionado anteriormente, a inferência consiste em : dado um objetivo(goal) checar se é possível derivá-lo com a kb definida

- Aqui, a inferência é uma função iterativa que aplica regras e coloca as novas conclusões na kb. Com essas novas conclusões, uma nova iteração e feita até (1) uma conclusão ser o goal ou (2) nao houver mais novas regras para concluir

Sintaxe - Exemplos

- Definindo dois Fatos : (define-facts (A B))

- Definindo a regra A or B => C : (define-rule (=> (or A B) C))

- Verifica se um novo D fato pode ser inferido : (infer D)

Exemplo

Código :

```
(define-facts (A B))

(define-rule (=> (and A B) C))

(define-rule (=> C D))

(define-rule (=> (or D E) F))

(infer F)
```

Passos :

0) fatos iniciais = (A B)

1) A and B -> C, fatos = (A B C)

2) C -> D, fatos (A B C D)

3) D or E -> F, fatos (A B C D F)

4) F está nos fatos

Update - Sintaxe

- A sintaxe foi mudada de algo LISP like para uma versão mais natural

- Definindo a regra A or B => C :
  - Antigo : (define-rule (=> (or A B) C))
  - **Atual : (rule A and B => C)**

Update - Sintaxe

- Agora, e possível definir funções e usá-las nas regras e inferência

- Ex :
- (define (menor x y)
- (< x y))
- (rule A **menor** B => C)

Update - Tratamento de variáveis e fatos

- Agora, fatos são armazenado como pares (símbolo . valor)
  Ex : Fatos Booleanos : (A #t)

  Ex : Fatos Numéricos : (B 5)

- A DSL permite inferência baseado em valores e não somente na presença

Inferência de uma Função

1) Verifica se : A premissa é uma lista cujo primeiro elemento é uma função definida no ambiente atual
2) Extrai o procedimento(função definida pelo usuário) e os argumentos
3) Percorre os argumentos e verifica se estão na KB, se eles tiverem um valor correspondente pega esse valor, se não usa o símbolo
4) aplica o procedimento aos valores(extraídos dos argumentos)

- Limitações : procedimentos com apenas 2 argumentos e, a função precisa retornar #t ou #f

Exemplo - Operador Booleano

(fact A) : Quando o fato é booleano, o fato é guardado como : (A #t)

(fact B)

```
(define (xor a b)
  (or (and a (not b)) (and b (not a))))
```

(rule A xor B => D)

(infer D) : Falso, pois A e B são True

Exemplo - Operador Aritmético

(fact Temp 35) : Quando defino o valor de forma explícita, o fato e numérico

(fact Limite 30)

(define (maior x y)

  (> x y))

(rule Temp maior Limite => quente)

(rule quente => alerta)

(infer alerta) : True, pois 35 > 30 e quente => alerta