lógica de 1ª ordem, inferência

lógica de 1ª ordem (recap.)

muito útil para expressar conhecimento

- matemática, filosofia, IA

permite expressar relações entre objetos

permite expressar factos e regras

- regras gerais (a todos os objetos)
- regras parciais (a alguns objetos)

é uma de várias linguagens formais de expressão de conhecimento:
- lógica proposicional
- lógica de 1ª ordem
- lógica temporal
- teoria de probabilidades
- lógica difusa

sintaxe de LPO

exemplo

$$
\begin{aligned}
\textit{Sentence} &\rightarrow \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} &\rightarrow \textit{Predicate} \mid \textit{Predicate}(\textit{Term}, \ldots) \mid \textit{Term} = \textit{Term} \\
\textit{ComplexSentence} &\rightarrow (\ \textit{Sentence}\ ) \\
&\mid \neg\ \textit{Sentence} \\
&\mid \textit{Sentence} \wedge \textit{Sentence} \\
&\mid \textit{Sentence} \vee \textit{Sentence} \\
&\mid \textit{Sentence} \Rightarrow \textit{Sentence} \\
&\mid \textit{Sentence} \Leftrightarrow \textit{Sentence} \\
&\mid \textit{Quantifier}\ \textit{Variable}, \ldots\ \textit{Sentence} \\
\textit{Term} &\rightarrow \textit{Function}(\textit{Term}, \ldots) \\
&\mid \textit{Constant} \\
&\mid \textit{Variable} \\
\textit{Quantifier} &\rightarrow \forall \mid \exists \\
\textit{Constant} &\rightarrow A \mid X_1 \mid \textit{John} \mid \cdots \\
\textit{Variable} &\rightarrow a \mid x \mid s \mid \cdots \\
\textit{Predicate} &\rightarrow \textit{True} \mid \textit{False} \mid \textit{After} \mid \textit{Loves} \mid \textit{Raining} \mid \cdots \\
\textit{Function} &\rightarrow \textit{Mother} \mid \textit{LeftLeg} \mid \cdots \\
\textsc{Operator Precedence} &: \neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow
\end{aligned}
$$

minúsculas

maiúsculas

símbolos – convenções e terminol.

termos

- constantes em minúscula
- variáveis em maiúscula

ao contrário da tabela anterior → para compatibilidade com Prolog

- predicados têm **aridade** = nº de argumentos

colega(*maria, manuel*) aridade 2

afirmações | asserções

atómicas

irmão(*manuel, joão*)

casada(*mãe*(*manuel*), *pai*(*joão*))

complexas

¬*irmão*(*brinquedo*(*manuel*), *joão*)

irmão(*manuel, joão*) ∧ *irmão*(*manuel, joão*)

ao_computador(*manuel*) ⇒ ¬*ao_computador*(*joão*)

leis de De Morgan

$$\forall X \ \neg p \equiv \neg \exists X \ \ p$$

$$\neg \forall X \ \ p \equiv \exists X \ \neg p$$

$$\forall X \ \ p \equiv \neg \exists X \ \neg p$$

$$\exists X \ \ p \equiv \neg \forall X \ \neg p$$

$$\neg(p \vee q) \equiv \neg p \wedge \neg q$$

$$\neg(p \wedge q) \equiv \neg p \vee \neg q$$

$$p \wedge q \equiv \neg(\neg p \vee \neg q)$$

$$p \vee q \equiv \neg(\neg p \wedge \neg q)$$

assunções semânticas

nomes únicos

cada constante refere-se a um objeto distinto

mundo fechado

afirmações atómicas não explicitadas são falsas
(só as afirmações explicitadas são verdadeiras)

domínio fechado

modelo não contém mais objetos do que as constantes definidas

base de conhecimento
knowledge base (KB)

KB - conjunto de factos e regras

asserções

$\textsc{Tell}(KB,\ aluna(maria))$.

$\textsc{Tell}(KB,\ pessoa(gervásio))$.

$\textsc{Tell}(KB,\ \forall X\ aluna(X) \Rightarrow pessoa(X))$.

interrogações

$\textsc{Ask}(KB,\ aluna(maria))$. devolve Verdade

$\textsc{Ask}(KB,\ pessoa(maria))$. devolve Verdade

$\textsc{Ask}(KB,\ pessoa(X))$. devolve $\{X/gervásio\}$ e $\{X/maria\}$

substituição

listas

para representar coleções

podem ter elementos repetidos, ao contrário dos conjuntos

[ ]	lista vazia
[x]	lista com o elemento x
[a, b, c]	…
[x | y]	lista com elemento x à cabeça e lista y na cauda

$\textsc{Tell}(KB, [a, b, c, d\,])$.

$\textsc{Ask}(KB, [X \mid Y\,]))$. devolve {$X/a$} e {$Y/[b, c, d\,]$}

a linguagem Prolog (facultativo)

inclui uma semântica operacional para representar conhecimento sob a forma de lógica de 1ª ordem

interpretador permite fornecer de 1 a todas as soluções de uma interrogação

inclui mecanismo de retropropagação para procura em árvore

em Prolog (facultativo)

$\textsc{Tell}(KB, [a, b, c, d])$. em Prolog: `assertz([a,b,c,d]).`

$\textsc{Ask}(KB, [X \mid Y]))$. em Prolog: `[X|Y].`

devolve: `X=a`

`Y=[b,c,d]`

```
$ gprolog
GNU Prolog 1.4.5 (64 bits)
Compiled Feb  5 2017, 10:30:08 with gcc
By Daniel Diaz
Copyright (C) 1999-2016 Daniel Diaz
| ?- assertz([a,b,c,d]).

yes
| ?- [X|Y].

X = a
Y = [b,c,d]

yes
| ?-
```

gnu prolog (gprolog)
para Linux, Windows

em Prolog (facultativo)

asserções

$\textsc{Tell}(KB,\ aluna(maria))$.

```
assertz(aluna(maria)).
```

$\textsc{Tell}(KB,\ pessoa(gervásio))$.

```
assertz(pessoa(gervásio)).
```

$\textsc{Tell}(KB,\ \forall X\ aluna(X) \Rightarrow pessoa(X))$.

```
assertz(pessoa(X):-aluna(X)).
```

interrogações

$\textsc{Ask}(KB,\ pessoa(X))$.

```
pessoa(X).
```

devolve:

```
X = gervásio ? ;
X = maria
```

dado pelo utilizador

engenharia do conhecimento

EC no 2º semestre da LEI

processo de construção de uma base de conhecimento num problema determinado

conhecimento específico

é um conjunto de 7 passos que estruturam esse processo

EC

1. *identificar a tarefa*

   qual o âmbito do problema a resolver? que tipo de conhecimento há disponível? que conhecimento representar?

2. *aquisição do conhecimento*

   obter o conhecimento dos peritos no problema; identificar a relevância do conhecimento

3. *decidir a ontologia do domínio*

   escolher o vocabulário de predicados, funções e constantes – uma teoria da natureza do que é relevante (no domínio do probl.)

EC

4. *codificar o conhecimento geral sobre o domínio*

   escrever os axiomas que definem os termos do vocabulário

5. *codificar uma descrição da instância específica do probl.*

   num agente, instâncias do problema são dadas pelos sensores

6. *fazer interrogações ao procedimento de inferência e obter respostas*

   inferência deriva factos que pretendemos saber

7. *depurar a base de conhecimento*

inferência em LPO

suponhamos a KB

$$\forall X \; jovem(X) \wedge pessoa(X) \Rightarrow estudante(X).$$
$$jovem(luna).$$
$$pessoa(luna).$$

inferência:
há alguma substituição de X por uma constante (na KB) que verifique as premissas? se sim, pode inferir-se o consequente, com a mesma substituição

neste caso a substituição θ = $\{X/luna\}$ verifica as premissas,

logo infere-se $estudante(luna)$

inferência em LPO

e agora com uma KB só relativa pessoas?

$$\forall X\ jovem(X) \wedge pessoa(X) \Rightarrow estudante(X).$$
$$jovem(luna).$$
$$pessoa(Y).$$

neste caso a substituição θ = $\{X/luna,\ Y/luna\}$ verifica as premissas,

logo infere-se $estudante(luna)$, também

modus ponens generalizado

resultado da substituição θ aplicada a p_i'

sendo p_i, p_i' e q afirmações atómicas com SUBST(θ,p_i)=SUBST(θ,p_i')

então

$$\frac{p_1', p_2', \ldots, p_n', \quad (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{\text{Subst}(\theta, q)}$$

dado infere-se

no caso do exemplo anterior

p_1' é $jovem(luna)$
p_2' é $pessoa(Y)$
θ é $\{X/luna, Y/luna\}$
$\text{Subst}(\theta, q)$ é $estudante(luna)$

p_1 é $jovem(X)$
p_2 é $pessoa(X)$
q é $estudante(X)$

unificação

é a operação de atribuir valores consistentes a variáveis

permite encontrar substituições adequadas para as inferências

definição

UNIFY (p, q) = θ, com SUBST (θ, p) = SUBST (θ, q).

exemplos (KB só com os dois factos nos argumentos de UNIFY)

UNIFY(*conhece*(*joão*, *X*), *conhece*(*joão*, *gina*)) = {*X*/*gina*}

UNIFY(*conhece*(*joão*, *X*), *conhece*(*Y*, *bruno*)) = {*X*/*bruno*, *Y*/*joão*}

UNIFY(*conhece*(*joão*, *X*), *conhece*(*Y*, *mãe*(*Y*))) = {*Y*/*joão*, *X*/*mãe*(*joão*)}

UNIFY(*conhece*(*joão*, *X*), *conhece*(*X*, *isabel*)) = fail. ← problema

variáveis separadas em afirmações separadas

UNIFY(*conhece*(*joão*, *X*), *conhece*(*X*, *isabel*)) = fail.

mas se as variáveis forem diferentes...

UNIFY(*conhece*(*joão*, *X*), *conhece*(X_{17}, *isabel*)) = {*X*/*isabel*, X_{17}/*joão*}

interrogações em prolog

ou em programação em lógica em geral

emprega(*ulisboa*, *antonio*)	a ULisboa emprega o Antonio?
emprega(*X*, *antonio*)	quem emprega o António?
emprega(*ulisboa*, *Y*)	quem é empregado da ULisboa?
emprega(*X*, *Y*)	quem emprega quem?