

Inteligência Artificial

Aula 4 - vídeo 2 - Negação por Falha

1 de setembro de 2020

Jogadores : $\{a, b, c, d\}$

Categorias: $\{vencedor, lutador, perdedor\}$

Fatos:

vencer(a,b).

vencer(c,a).

vencer(d,b).

Perguntas: Quem são vencedores, lutadores e perdedores ?

Fatos:

vencer(a,b).
vencer(c,a).
vencer(d,b).

Perguntas: Quem são vencedores, lutadores e perdedores ?

Se X venceu alguém e alguém venceu de X
então X é um lutador
c.c. se X venceu alguém
então X é um vencedor
c.c. X é um perdedor

Fatos:

vencer(a,b).

vencer(c,a).

vencer(d,b).

Perguntas: Quem são vencedores, lutadores e perdedores ?

Fatos:

```
vencer(a,b).  
vencer(c,a).  
vencer(d,b).
```

Perguntas: Quem são vencedores, lutadores e perdedores ?

```
categ(X, lutador) :- vencer(X, _), vencer(_, X), !.
```

Fatos:

```
vencer(a,b).  
vencer(c,a).  
vencer(d,b).
```

Perguntas: Quem são vencedores, lutadores e perdedores ?

```
categ(X, lutador) :- vencer(X, _), vencer(_, X), !.  
categ(X, vencedor) :- vencer(X, _), !.
```

Fatos:

```
vencer(a,b).  
vencer(c,a).  
vencer(d,b).
```

Perguntas: Quem são vencedores, lutadores e perdedores ?

```
categ(X, lutador) :- vencer(X, _), vencer(_, X), !.  
categ(X, vencedor) :- vencer(X, _), !.  
categ(X, perdedor) :- vencer(_, X).
```

Programa:

```
vencer(a,b).  
vencer(c,a).  
vencer(d,b).  
categ(X,lutador) :- vencer(X,_), vencer(_,X), !.  
categ(X,vencedor) :- vencer(X,_), !.  
categ(X,perdedor) :- vencer(_,X).
```


Programa:

```
vencer(a,b).  
vencer(c,a).  
vencer(d,b).  
categ(X,lutador) :- vencer(X,_), vencer(_,X), !.  
categ(X,vencedor) :- vencer(X,_), !.  
categ(X,perdedor) :- vencer(_,X).
```

Como são respondidas as consultas?

```
?- categ(a,lutador).  
?- categ(a,vencedor).  
?- categ(a,perdedor).
```

Programa:

```
vencer(a,b).  
vencer(c,a).  
vencer(d,b).  
categ(X,lutador) :- vencer(X,_), vencer(_,X), !.  
categ(X,vencedor) :- vencer(X,_), !.  
categ(X,perdedor) :- vencer(_,X).
```

Como são respondidas as consultas?

```
?- categ(a,lutador).  
?- categ(a,vencedor).  
?- categ(a,perdedor).
```

Todas tem resposta **true**

Negação por Falha

$\text{not}(P)$ ou $\neg(P)$

Negação por Falha

$\text{not}(P)$ ou $\neg(P)$

Se P é bem sucedido
então $\text{not}(P)$ falha ;
c.c. $\text{not}(P)$ é bem sucedido.

Negação por Falha

$\text{not}(P)$ ou $\neg(P)$

Se P é bem sucedido
então $\text{not}(P)$ falha ;
c.c. $\text{not}(P)$ é bem sucedido.

Traduzindo para Prolog:


```
not(P) :- P, !, fail.  
not(P).
```

Negação por Falha

Exemplo

```
1 not(P) :- P, !, fail.  
2 not(P).  
3 q(a).
```

?- not(q(a)).

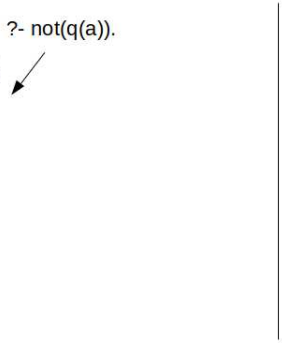


Negação por Falha

Exemplo

```
1 not(P) :- P, !, fail.  
2 not(P).  
3 q(a).
```

1:P/q(a) ?- not(q(a)).



The diagram illustrates a query and a clause. On the left, the clause is labeled '1:P/q(a)'. On the right, the query is labeled '?- not(q(a)).'. A vertical line separates the two. An arrow points from the query to the clause, indicating a relationship or a step in a process.

Negação por Falha

Exemplo

```
1 not(P) :- P, !, fail.  
2 not(P).  
3 q(a).
```

?- not(q(a)).
1:P/q(a) ↙
?- q(a), !, fail.

Negação por Falha

Exemplo

```
1 not(P) :- P, !, fail.  
2 not(P).  
3 q(a).
```

?- not(q(a)).
1:P/q(a) ↙
?- q(a), !, fail.
↓
?- !, fail.

Negação por Falha

Exemplo

```
1 not(P) :- P, !, fail.  
2 not(P).  
3 q(a).
```

?- not(q(a)).

1:P/q(a) ↙

?- q(a), !, fail.

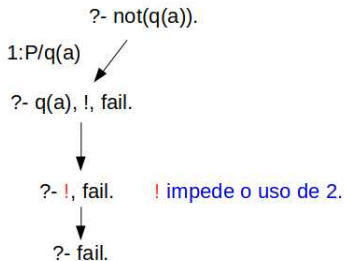
↓

?- !, fail. ! impede o uso de 2.

Negação por Falha

Exemplo

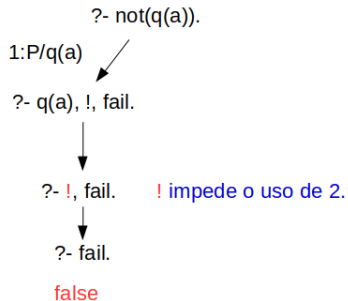
```
1 not(P) :- P, !, fail.  
2 not(P).  
3 q(a).
```



Negação por Falha

Exemplo

```
1 not(P) :- P, !, fail.  
2 not(P).  
3 q(a).
```



Negação por Falha

Exemplo

```
1 not(P) :- P, !, fail.  
2 not(P).  
3 q(a).
```

?- not(q(a)).

1:P/q(a) ↙

?- q(a), !, fail.

↓

?- !, fail. ! impede o uso de 2.

↓

?- fail.

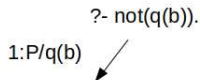
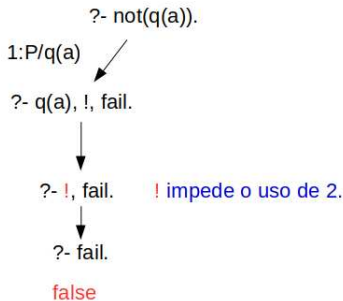
false

?- not(q(b)).

Negação por Falha

Exemplo

```
1 not(P) :- P, !, fail.  
2 not(P).  
3 q(a).
```



Negação por Falha

Exemplo

```
1 not(P) :- P, !, fail.  
2 not(P).  
3 q(a).
```

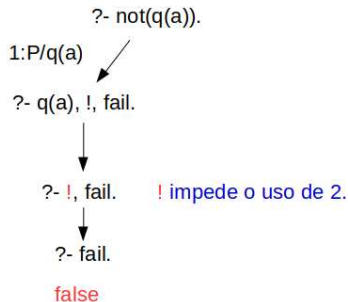


Diagram illustrating the execution of the Prolog query `?- not(q(b)).` using the provided rules.

The query `?- not(q(b)).` is processed by rule 1, which is `not(P) :- P, !, fail.` This results in the goal `?- q(b), !, fail.`

The goal `?- q(b), !, fail.` is processed by rule 3, which is `q(a).` This results in the goal `?- !, fail.`

The goal `?- !, fail.` is processed by rule 2, which is `not(P).` This results in the goal `?- fail.`

The final result is **false**.

Negação por Falha

Exemplo

```
1 not(P) :- P, !, fail.  
2 not(P).  
3 q(a).
```

?- not(q(a)).

1:P/q(a) ↘

?- q(a), !, fail.

↓

?- !, fail. ! impede o uso de 2.

↓

?- fail.

false

?- not(q(b)).

1:P/q(b) ↘

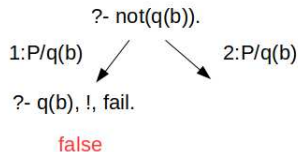
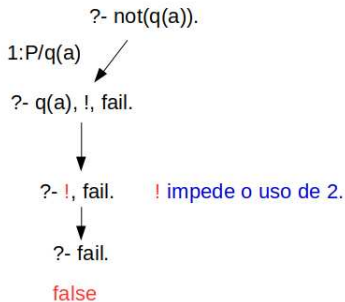
?- q(b), !, fail.

false

Negação por Falha

Exemplo

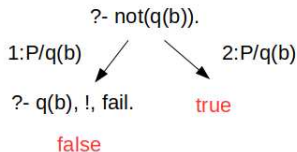
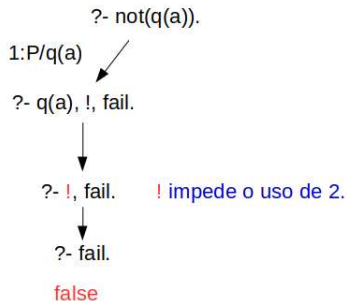
```
1 not(P) :- P, !, fail.  
2 not(P).  
3 q(a).
```



Negação por Falha

Exemplo

```
1 not(P) :- P, !, fail.  
2 not(P).  
3 q(a).
```



Negação por Falha

`not(P)` ou `\+(P)`

Se P é bem sucedido
então `not(P)` falha ;
c.c. `not(P)` é bem sucedido.

`not(P) :- P, !, fail.`
`not(P).`

`vencer(a,b).`
`vencer(c,a).`
`vencer(d,b).`

Negação por Falha

`not(P)` ou `\+(P)`

Se P é bem sucedido
então `not(P)` falha ;
c.c. `not(P)` é bem sucedido.

`not(P) :- P, !, fail.`
`not(P).`

`vencer(a,b).`
`vencer(c,a).`
`vencer(d,b).`

`categ(X, lutador) :- vencer(X, _), vencer(_, X).`

Negação por Falha

`not(P)` ou `\+(P)`

Se P é bem sucedido
então `not(P)` falha ;
c.c. `not(P)` é bem sucedido.

`not(P) :- P, !, fail.`
`not(P).`

`vencer(a,b).`
`vencer(c,a).`
`vencer(d,b).`

`categ(X, lutador) :- vencer(X, _), vencer(_, X).`
`categ(X, vencedor) :- vencer(X, _), not(vencer(_, X)).`

Negação por Falha

`not(P)` ou `\+(P)`

Se P é bem sucedido
então `not(P)` falha ;
c.c. `not(P)` é bem sucedido.

`not(P) :- P, !, fail.`
`not(P).`

`vencer(a,b).`
`vencer(c,a).`
`vencer(d,b).`

`categ(X, lutador) :- vencer(X, _), vencer(_, X).`
`categ(X, vencedor) :- vencer(X, _), not(vencer(_, X)).`
`categ(X, perdedor) :- vencer(_, X), not(vencer(X, _)).`

Inteligência Artificial

Aula 4 - vídeo 2 - Negação por Falha

1 de setembro de 2020