



## Prolog lecture 6

Go to:

<http://etc.ch/7ctV>

Or scan the  
barcode

# Today's discussion

## Videos

Countdown

Graph search

Q: When figuring out what a Prolog program does, how can we work out which of the arguments are intended to be supplied with constants, and which with variables.

A: Did I manage to answer this last time?

Q: What does Prolog allow us to do (other than coding in a different way) that other languages can't?  
Not meaning to sound dismissive just curious of applications!

Q: What does Prolog allow us to do (other than coding in a different way) that other languages can't?  
Not meaning to sound dismissive just curious of applications!

A: I'll try and include some examples in the lectures

# Example: Datomic

A database system

- indelibility and auditability
- a flexible data model
- ACID transactions
- a powerful query language
- and horizontal read scaling



# Datomic

A database system

- indelibility and auditability
- a flexible data model
- ACID transactions
- a powerful query language
- and horizontal read scaling

← Datalog (simple Prolog)



# Datomic



;;which 42-year-olds like what?

```
[ :find ?e ?x
```

```
  :where [?e :age 42] [?e :likes ?x]]
```



# Datomic



;;which 42-year-olds like what?

```
[ :find ?e ?x
```

```
  :where [?e :age 42] [?e :likes ?x]]
```

```
find(E,X) :- age(E,42), likes(E,X).
```

Q: you generally put the base case rule first e.g.  
Split([], [], []) - wouldn't it be more efficient to put this  
last since it is less likely? (fewer unifications)

Q: you generally put the base case rule first e.g. `Split([], [], [])` - wouldn't it be more efficient to put this last since it is less likely? (fewer unifications)

A: you would make a small saving if you only wanted one answer but more answers were possible. But you would still have all the choice points. Remember that order often matters when you have cut.

Q: Do we need to be able to compare Prolog to ML and functional programming? As a third year 50%er that was all a while ago...

Q: Do we need to be able to compare Prolog to ML and functional programming? As a third year 50%er that was all a while ago...

A: I won't ask you to write ML in the exam. (But I would expect you to recall the concepts of the ML course as a general principle - what's the point of your degree otherwise?)

Q: You mentioned that we can use cuts and negation in the exam. Can we also use implication ( $\rightarrow$ )?

Q: You mentioned that we can use cuts and negation in the exam. Can we also use implication ( $\rightarrow$ )?

A: No. You also can't use `;`, assume any library predicates, or use any extra-logical stuff (except cut) like `findAll`, `call` etc.

Q: What is the underlying difference between a rule and a compound term? Same syntax right?



Q: What is the underlying difference between a rule and a compound term? Same syntax right?

A: a compound term is a 'term' in first order logic, a rule is 'formula' in first order logic.

## Definition 3

The terms  $t, u, \dots$  of a first-order language are defined recursively as follows:

- A variable is a term.
- A constant symbol is a term.
- If  $t_1, \dots, t_n$  are terms and  $f$  is an  $n$ -place function symbol then  $f(t_1, \dots, t_n)$  is a term.

## Definition 4

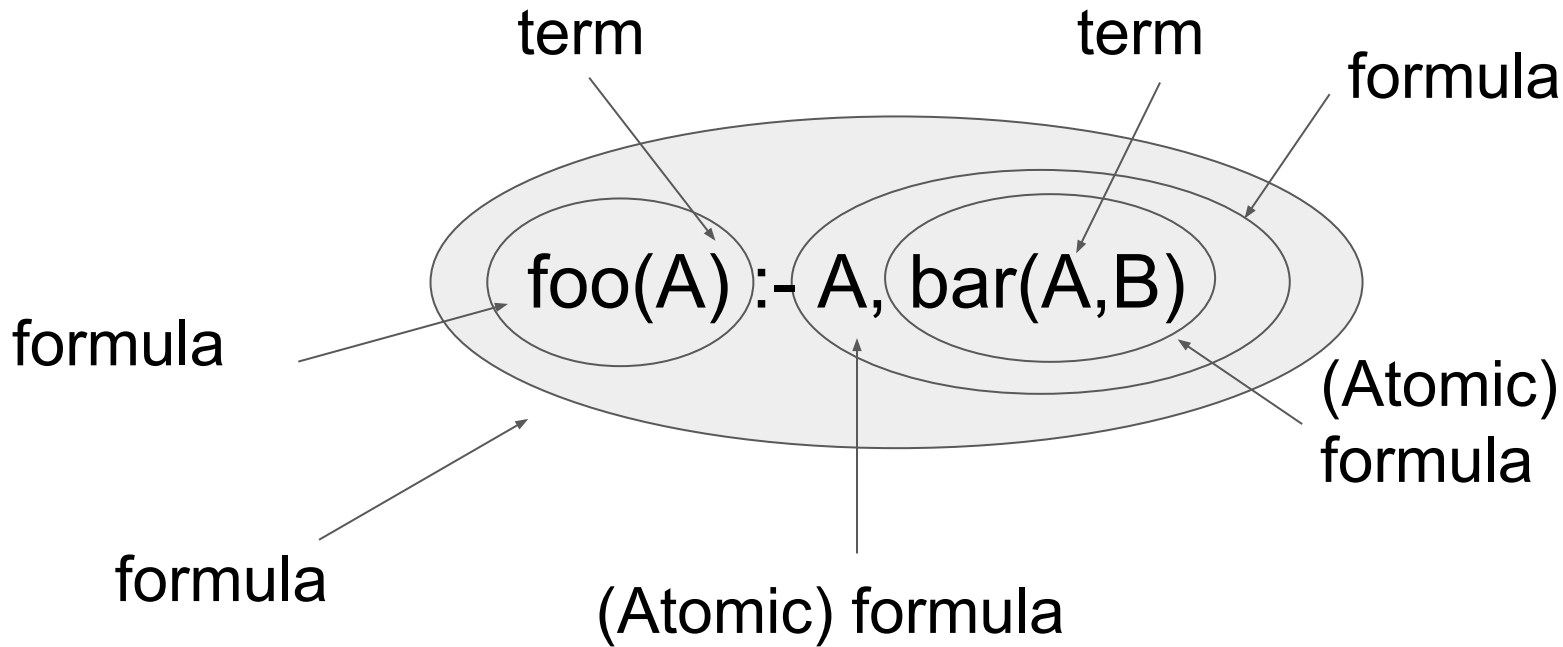
The formulas  $A, B, \dots$  of a first-order language are defined recursively as follows:

- If  $t_1, \dots, t_n$  are terms and  $P$  is an  $n$ -place predicate symbol then  $P(t_1, \dots, t_n)$  is a formula
- If  $A$  and  $B$  are formulas then  $\neg A, A \wedge B, A \vee B, A \rightarrow B, A \leftrightarrow B$  are also formulas.
- If  $x$  is a variable and  $A$  is a formula then  $\forall x A$  and  $\exists x A$  are also formulas.

$\forall A((\exists B(A \wedge \text{bar}(A, B))) \implies \text{foo}(A))$



$\text{foo}(A) \text{ :- } A, \text{bar}(A, B)$

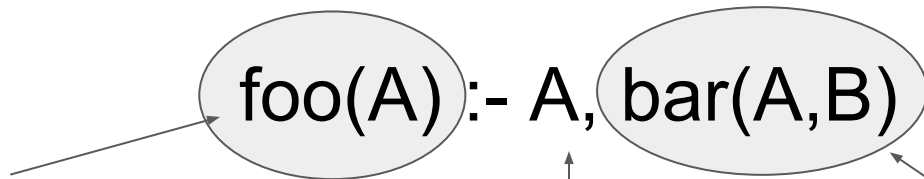


term

term

foo(A) :- A, bar(A,B)

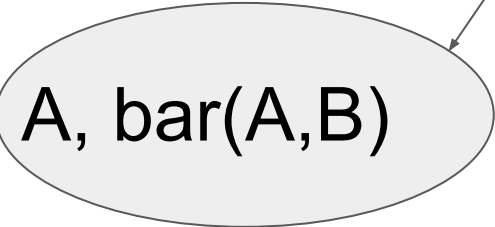
Atomic  
formula



(Atomic) formula

(Atomic)  
formula

foo(A) :- A, bar(A,B)



formula

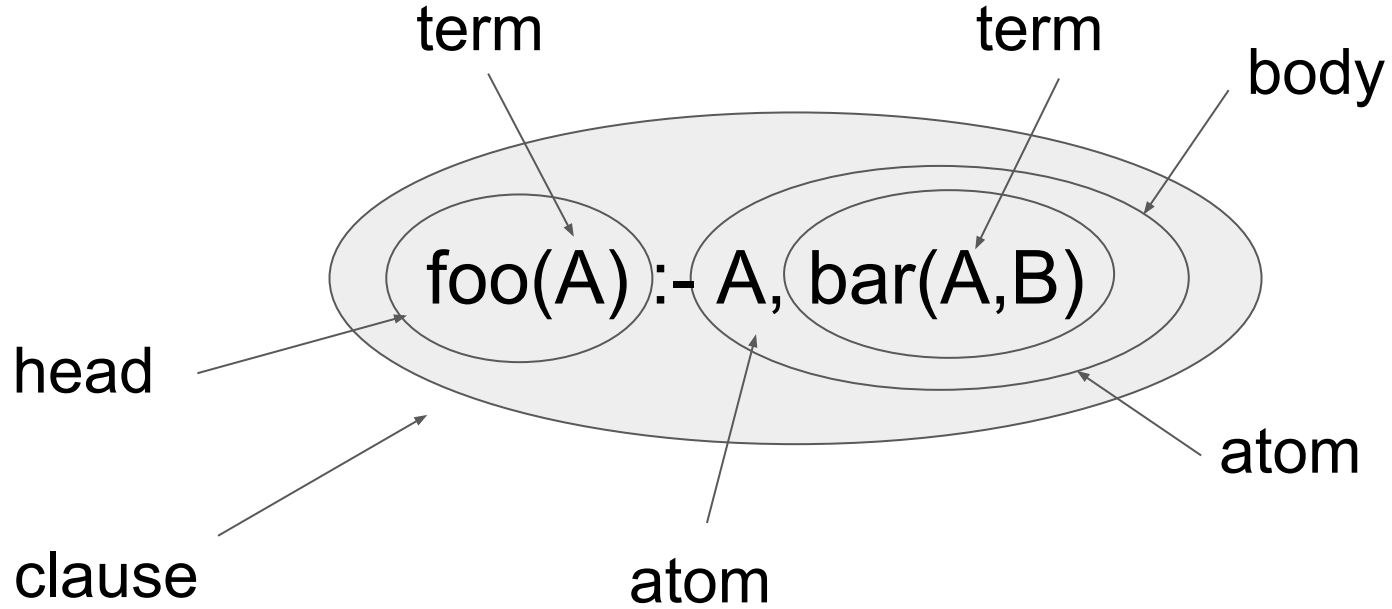




foo(A) :- A, bar(A,B)

formula

Back to prolog....



Q: Is single cut rule bad practice?  $\text{Ist}(H, [H])$ .  
 $\text{Ist}(X, \_ | T) :- \text{Ist}(X, T)$ . This pointlessly backtracks after finding the answer. So change axiom to:  $\text{Ist}(H, [H]) :-$   
!

Q: Is single cut rule bad practice?  $\text{Ist}(H,[H])$ .  
 $\text{Ist}(X,[_|T]):-\text{Ist}(X,T)$ . This pointlessly backtracks after finding the answer. So change axiom to:  $\text{Ist}(H,[H]) :- !$ .

A: Its fine to put a cut on an axiom. The! Thing! To!  
Avoid! Is! Putting! One! Everywhere!

## A bigger cut example...

a(1).

a(a).

b(3).

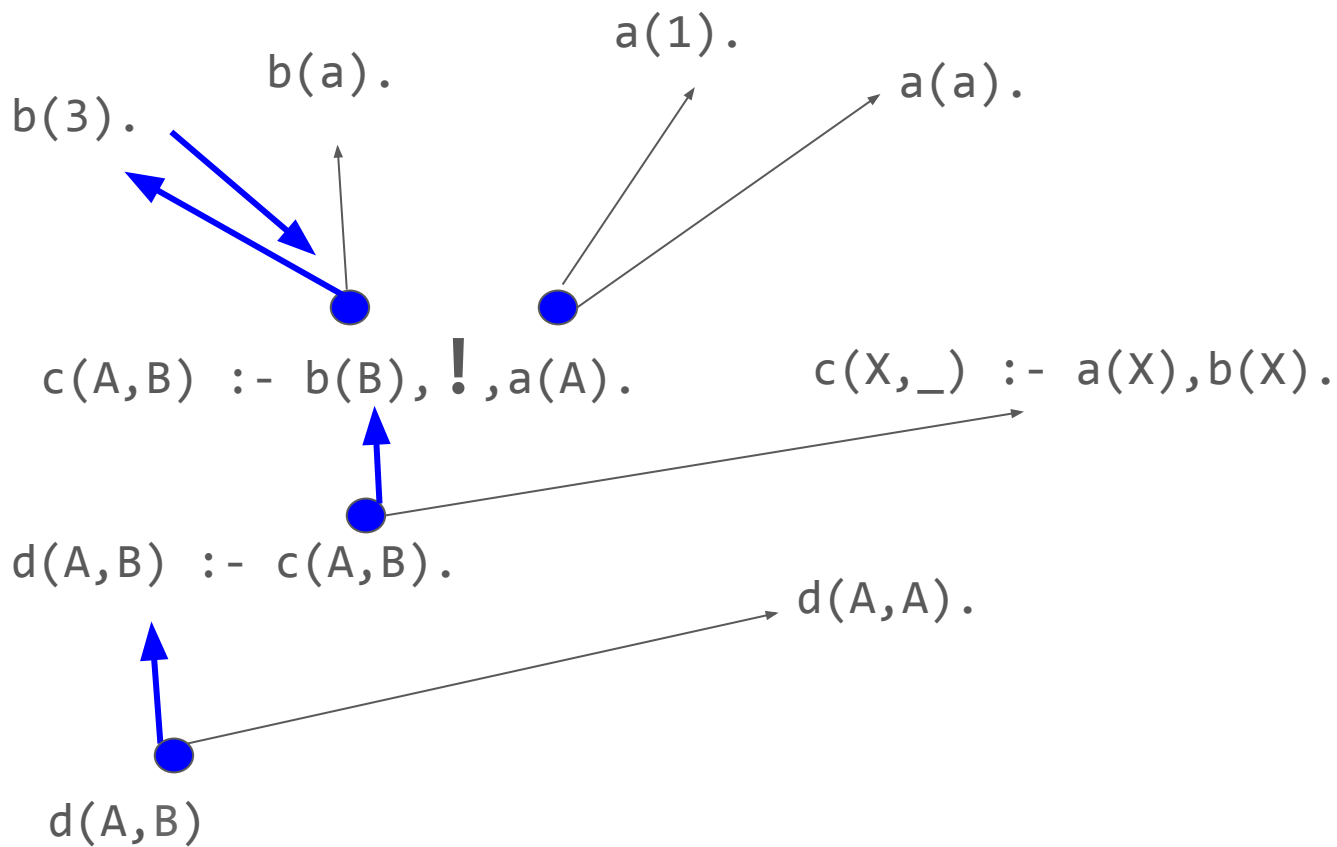
b(a).

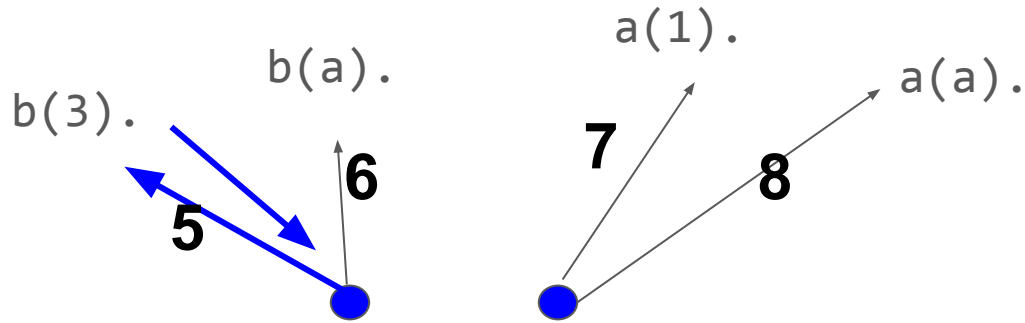
c(A,B) :- b(B),!,a(A).

c(X,\_) :- a(X),b(X).

d(A,B) :- c(A,B).

d(A,A).



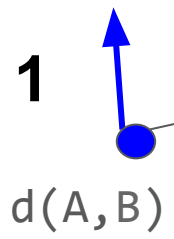


$c(A,B) :- b(B), !, a(A).$

$c(X,_) :- a(X), b(X).$

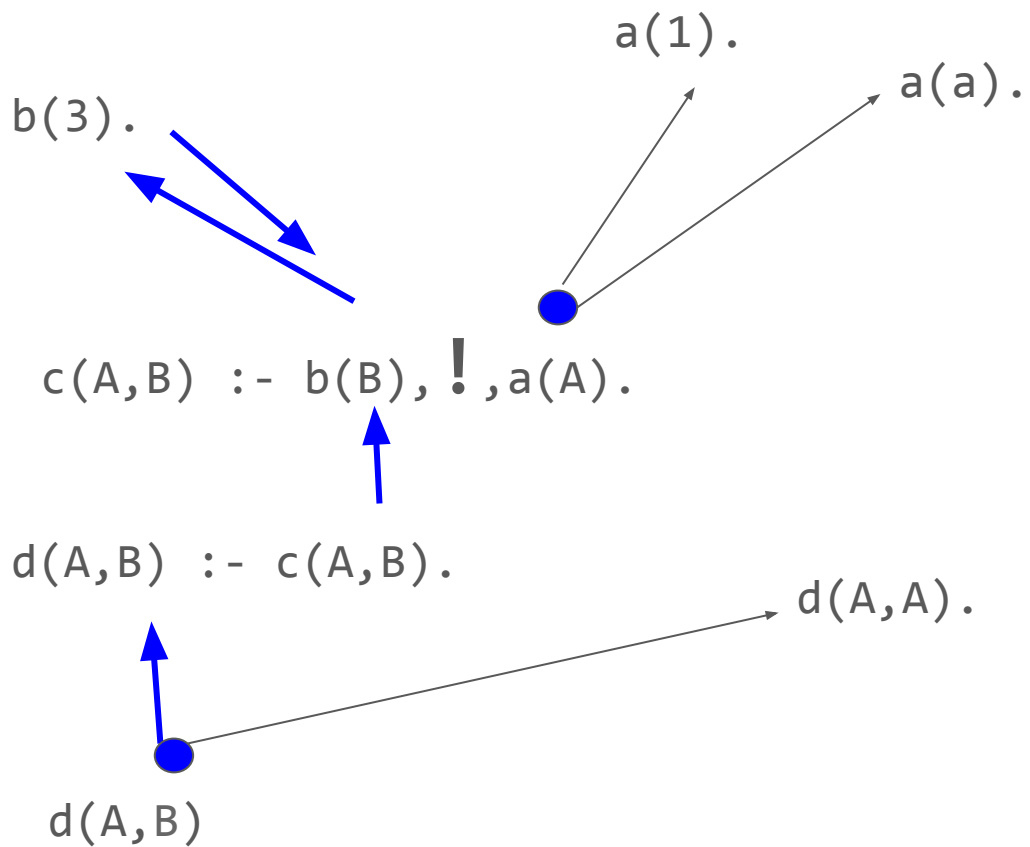
$d(A,B) :- c(A,B).$

$d(A,A).$



2

Which of these edges are removed by crossing the cut?



```
?- d(A,B).
A=1, B=3 ;
A=a, B=3 ;
A=B.
```



# Challenge: Write a tic-tac-toe (noughts and crosses) AI

What's the first step?

# Next time

## Videos

Difference

Empty difference lists

Difference list example

Ask questions at [www.slido.com](http://www.slido.com) with event code E508