CSc 520

Principles of Programming Languages

43 : Logic Programming — Prolog Unification

Christian Collberg

collberg+520@gmail.com

Department of Computer Science

University of Arizona

Unification & Matching

- So far, when we've gone through examples, I have said simply that when trying to satisfy a goal, Prolog searches for a matching rule or fact.
- What does this mean, to match?
- Prolog's matching operator or =. It tries to make its left and right hand sides the same, by assigning values to variables.
- Also, there's an implicit = between arguments when we try to match a query

```
?- f(x,y)
```

to a rule

```
f(A,B) :- ....
```

Matching Examples

The rule:

```
deriv(U ^C, X, C * U ^L * DU) :-
        number(C), L is C - 1,
        deriv(U, X, DU).
```

```
?- deriv(x ^3, x, D).
   D = 1*3*x^2
```

The goal:

- `x ^3` matches `U ^C`
  - $x = U, C = 3$
- `x` matches `X`
- `D` matches `C * U ^L * DU`

Matching Examples. . .

```
deriv(U+V, X, DU + DV) :-
    deriv(U, X, DU),
    deriv(V, X, DV).

?- deriv(x^3 + x^2 + 1, x, D).
    D = 1*3*x^2+1*2*x^1+0
```

- `x ^3 + x^2 + 1` matches `U + V`
  - `x ^3 + x^2` is bound to `U`
  - `1` is bound to `V`

Matching Algorithm

Can two terms A and F be "made identical," by assigning values to their variables?

Two terms A and F match if

1. they are identical atoms
2. one or both are uninstantiated variables
3. they are terms $A = f_A(a_1, \cdots, a_n)$ and $F = f_F(f_1, \cdots, f_m)$, and
   (a) the arities are the same ($n = m$)
   (b) the functors are the same ($f_A = f_F$)
   (c) the arguments match ($a_i \equiv f_i$)

Matching – Examples

A	F	$A \equiv F$	variable subst.
a	a	yes	
a	b	no	
sin(X)	sin(a)	yes	$\theta = \{$X=a$\}$
sin(a)	sin(X)	yes	$\theta = \{$X=a$\}$
cos(X)	sin(a)	no	
sin(X)	sin(cos(a))	yes	$\theta = \{$X=cos(a)$\}$

Matching – Examples...

A	F	$A \equiv F$	variable subst.
likes(c, X)	likes(a, X)	no	
likes(c, X)	likes(c, Y)	yes	$\theta = \{$X=Y$\}$
likes(X, X)	likes(c, Y)	yes	$\theta = \{$X=c, X=Y$\}$
likes(X, X)	likes(c, _)	yes	$\theta = \{$X=c, X=_47$\}$
likes(c, a(X))	likes(V, Z)	yes	$\theta = \{$V=c,Z=a(X)$\}$
likes(X, a(X))	likes(c, Z)	yes	$\theta = \{$X=c,Z=a(X)$\}$

Matching Consequences

Consequences of Prolog Matching:

- An uninstantiated variable will match any object.
- An integer or atom will match only itself.
- When two uninstantiated variables match, they *share*:
  - When one is instantiated, so is the other (with the same value).
- Backtracking undoes all variable bindings.

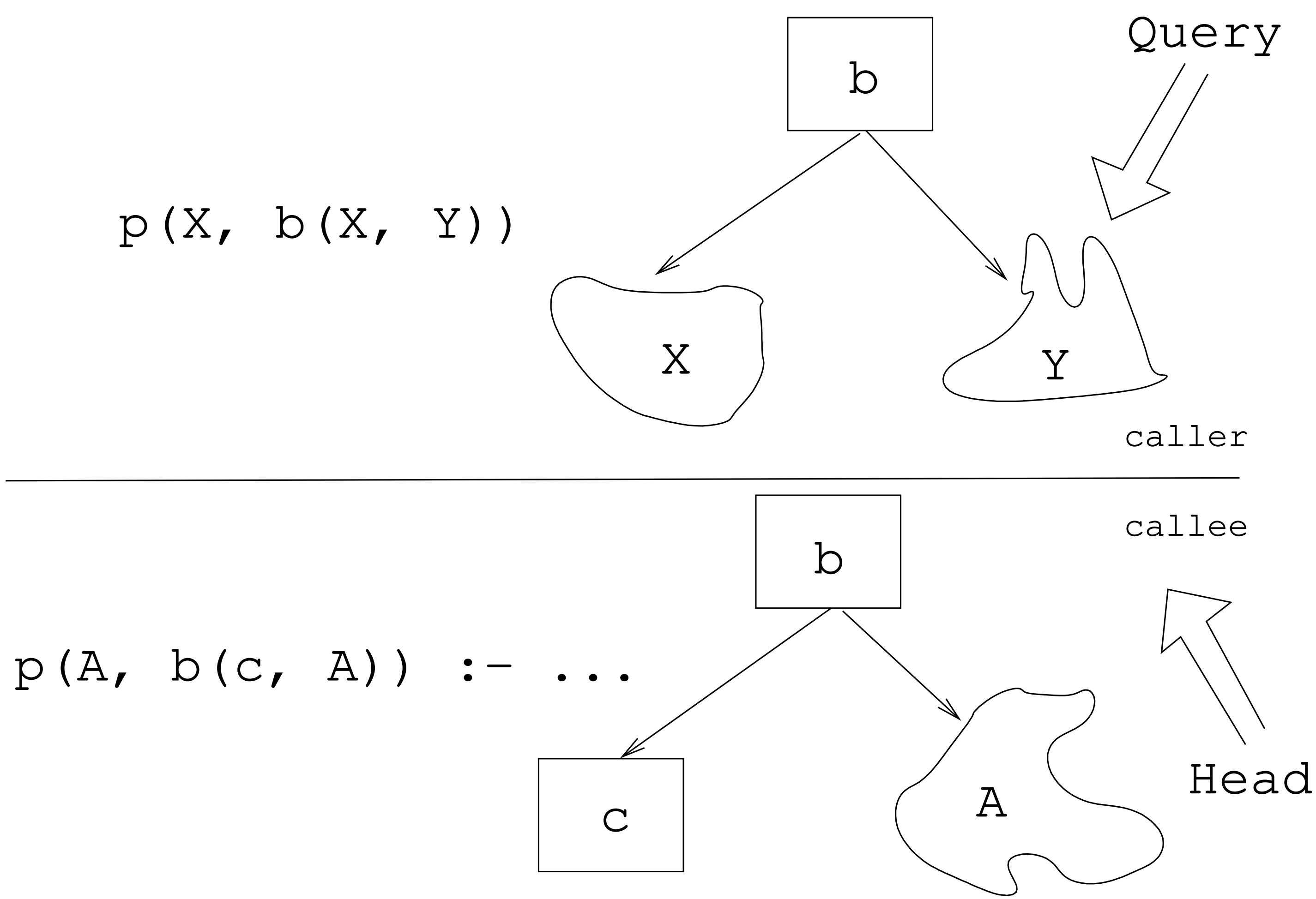
Matching Algorithm

```
FUNC Unify (A, F: term) :  BOOL;
    IF Is_Var(F) THEN Instantiate F to A
    ELSIF Is_Var(A) THEN Instantiate A to F
    ELSIF Arity(F)≠Arity(A) THEN RETURN FALSE
    ELSIF Functor(F)≠Functor(A) THEN RETURN FALSE
    ELSE
        FOR each argument i DO
            IF NOT Unify(A(i), F(i)) THEN
                RETURN FALSE
    RETURN TRUE;
```

Visualizing Matching

- From *Prolog for Programmers*, Kluzniak & Szpakowicz, page 18.
- Assume that during the course of a program we attempt to match the goal `p(X, b(X, Y))` with a clause C, whose head is `p(X, b(X, y))`.
- First we'll compare the arity and name of the functors. For both the goal and the clause they are `2` and `p`, respectively.

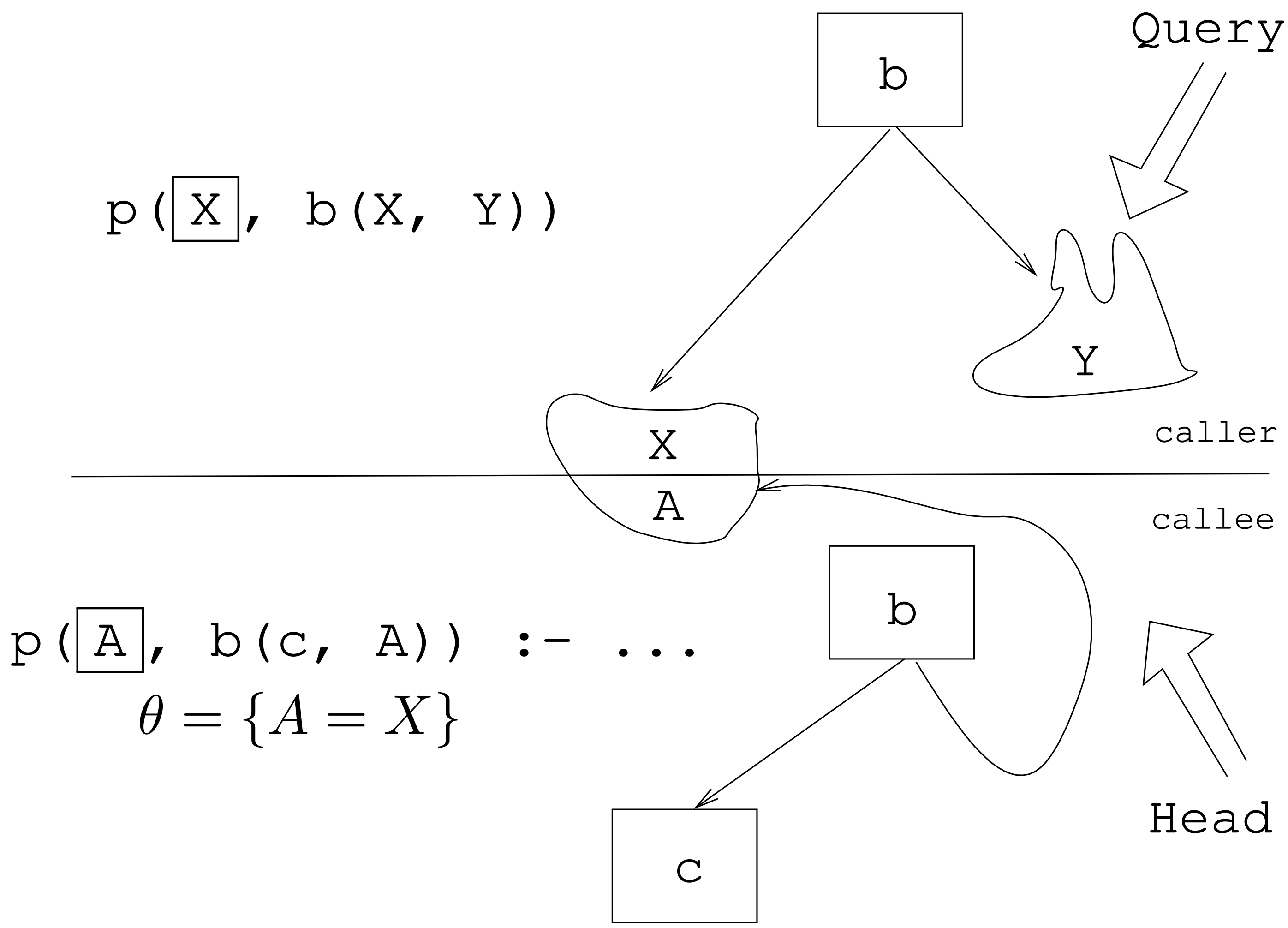
Visualizing Matching. . .



Visualizing Matching. . .

- The second step is to try to unify the first argument of the goal (`X`) with the first argument of the clause head (`A`).
- They are both variables, so that works OK.
- From now on `A` and `X` will be treated as identical (they are in the list of variable substitutions θ).

Visualizing Matching. . .

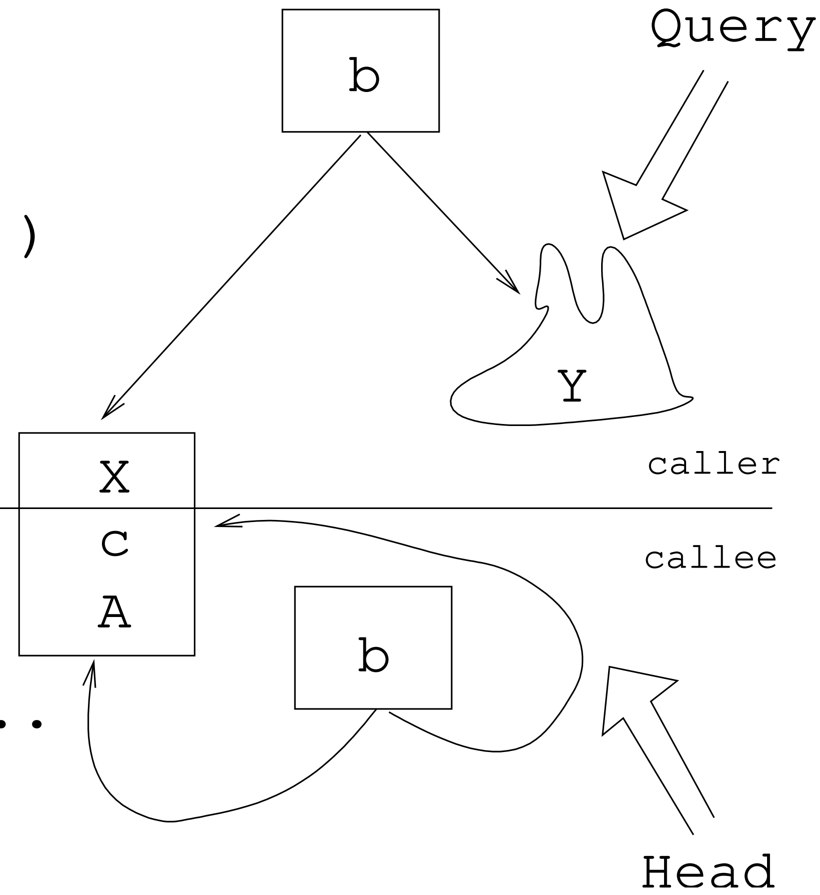


Visualizing Matching. . .

- Next we try to match the second argument of the goal (`b(X, Y)`) with the second argument of the clause head (`b(c, A)`).
- The arities and the functors are the same, so we go on to to try to match the arguments.
- The first argument in the goal is `X`, which is matched by the first argument in the clause head (`c`). I.e., `X` and `c` are now treated as identical.

Visualizing Matching...

`p(X, b(`$\boxed{\texttt{X}}$`, Y))`



`p(A, b(`$\boxed{\texttt{c}}$`, A)) :- ...`

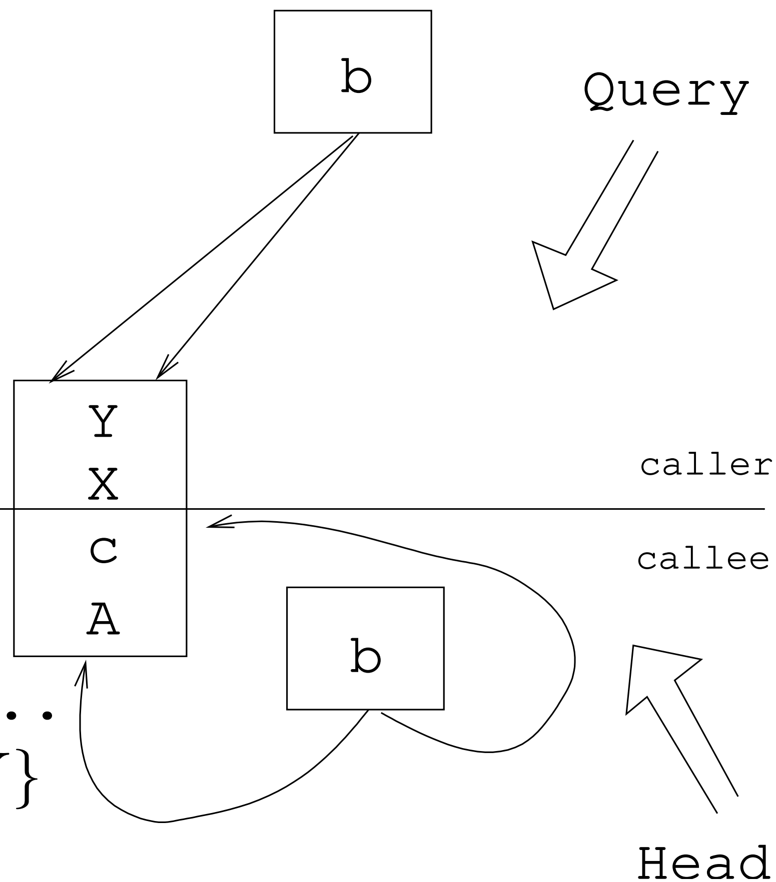
$$\theta = \{A = X, X = c\}$$

Visualizing Matching. . .

- Finally, we match `A` and `Y`. Since `A=X` and `X=c`, this means that `Y=c` as well.

Visualizing Matching…

`p(X, b(X, `$\boxed{\texttt{Y}}$`))`



`p(A, b(c, `$\boxed{\texttt{A}}$`)) :- ...`

$$\theta = \{A = X, X = c, A = Y\}$$

Readings and References

- Read Clocksin-Mellish, Sections 2.4, 2.6.3.

Prolog So Far...

- A term is either a
  - a constant (an atom or integer)
  - a variable
  - a structure
- Two terms *match* if
  - there exists a variable substitution θ which makes the terms identical.
- Once a variable becomes instantiated, it stays instantiated.
- Backtracking *undoes* variable instantiations.
- Prolog searches the database sequentially (from top to bottom) until a matching clause is found.