Week 10

Brendon Faleiro

Prolog

- In order to solve a prolog query, the prolog engine builds the search tree based on the provided facts and rules.

- It then runs a depth first search on this tree.

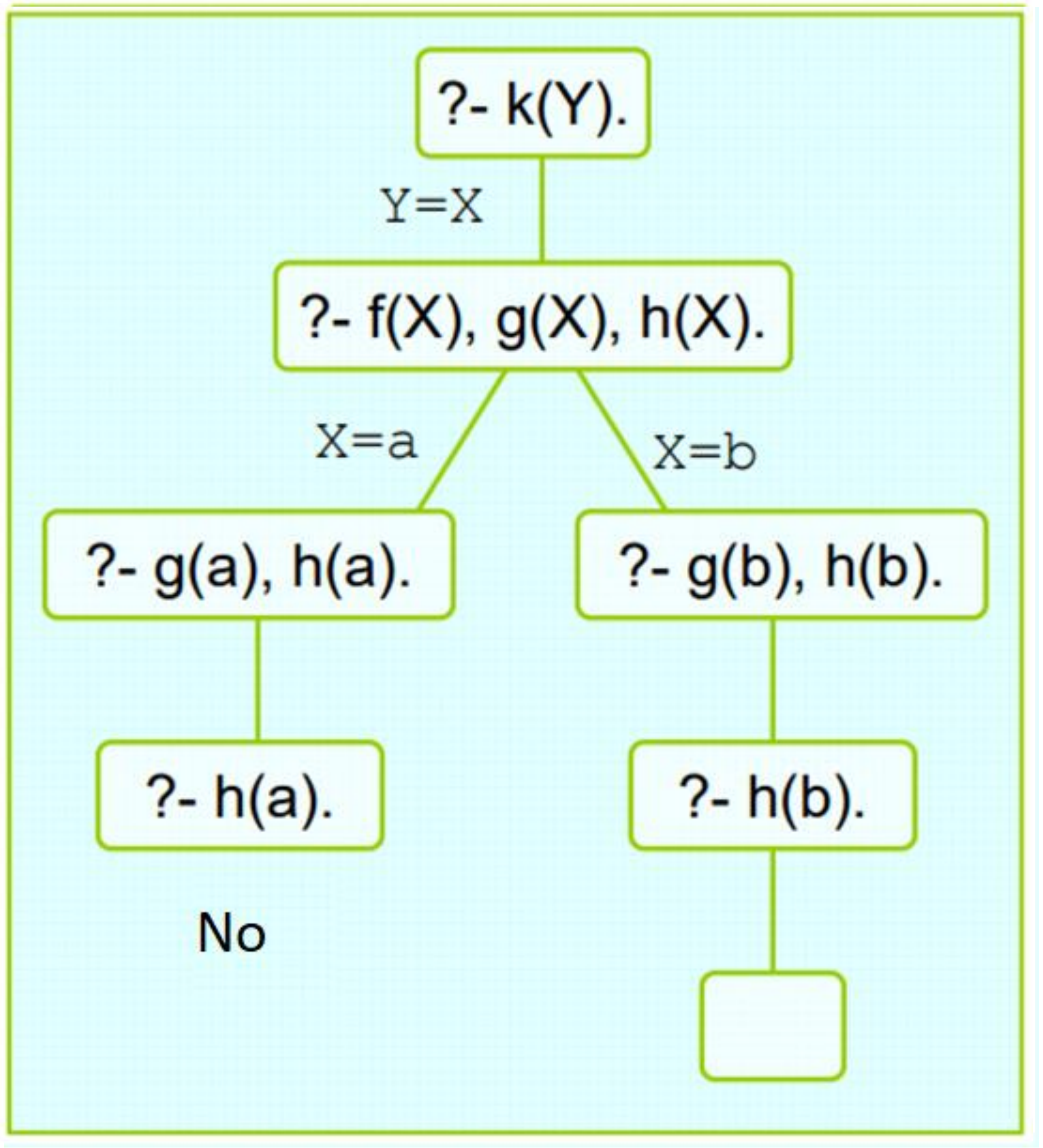
- While performing the search, Prolog uses unification to match terms and map variables to terms.

- Unification takes two terms t1 and t2 and returns either NO or an environment (mapping variables to terms) that makes t1 and t2 syntactically identical.

Example:

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

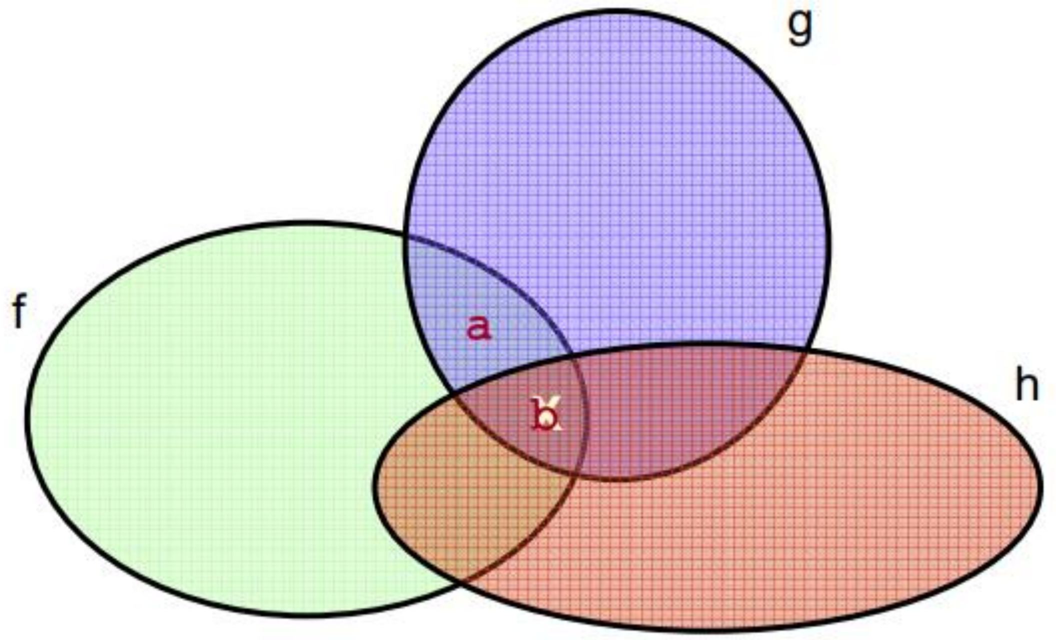
```
?- k(Y).
Y = b
```



- Source: http://www.fb10.uni-bremen.de/anglistik/ling/ws08/intro-cl-materials/prolog-search-tree.pdf

What problem did we just solve?

- f(a).
- f(b).
- g(a).
- g(b).
- h(b).



Example for practice:

- mystery([],[]).
- mystery([b,a|T1],[z,z|T2]) :- mystery(T1,T2).
- mystery([X|T1],[X|T2]) :- mystery(T1,T2).

- Construct a tree for the following query:

? – mystery([b,a,b], Z).

- Write implementations of the following Prolog predicates.

- shift_left(L, R) succeeds if R is the result of "shifting left" the list L by 1. The leading element of L is lost. For example, shift_left([a,b,c], [b,c]).

Ans:
```
shift_left([_|R], R).
```

- shift_right(L, R) is similar, except it shifts right. For example, shift_right([a,b,c], [a,b]).

Ans:
```
shift_right([_], []).
shift_right([H|L], [H|R]) :- shift_right(L,R).
```

- shift_left_circular(L, R) is like shift_left, except the leading element of L is reintroduced at the right. For example, shift_left_circular([a,b,c], [b,c,a]).

Ans:
```
shift_left_circular([H|L], R) :- append(L, [H], R).
```

- shift_right_circular(L, R) is similar, except it shifts right. For example, shift_right_circular([a,b,c], [c,a,b]).

Ans:
```
shift_right_circular(L,R) :- shift_left_circular(R,L).
```

Revision - OCaml

- Recursion
- Higher-order functions
- Datatypes
- Pattern matching
- Exceptions

Java

- Subtyping
  - Any class that implements a certain interface or extends a particular class is said to be a subtype of that class.
  - A subtype can do everything that a parent type can do and more.
  - You can always pass a subtype wherever a parent type is wanted.
- Inheritance
  - It is used to avoid code duplication by making code from the parent class available in the child class.

- Dynamic dispatch
  - Dynamic dispatch only looks at the receiver object. It never looks at the argument.
  - Based on the type of the receiver object at run time appropriate functions are called.
  - This is how Java achieves method overriding.
  - When an overridden method is called through a superclass reference, Java determines which version(superclass/subclasses) of that method is to be executed based upon the type of the object being referred to at the time the call occurs. Thus, this determination is made at run time.
  - At run-time, it depends on the type of the object being referred to (not the type of the reference variable) that determines which version of an overridden method will be executed
  - A superclass reference variable can refer to a subclass object. This is also known as upcasting. Java uses this fact to resolve calls to overridden methods at run time.

- Static overloading
  - At compile time, the compiler binds the function call based on the function signature (function parameters).
  - Since this happens at compile time, the compiler does not check consider the runtime type of the parameter.

- Memory model in Java
  - Object storage
  - Aliasing
- parameter passing
  - by value vs. by reference
- parametric polymorphism (generics)
- exceptions, including throws annotations
- parallelism: fork/join, streams

Questions???

That's all Folks!
kalilak