Prolog programming: a do-it-yourself course for beginners

Day 1

Kristina Striegnitz

Department of Computational Linguistics

Saarland University, Saarbrücken, Germany

kris@coli.uni-sb.de

http://www.coli.uni-sb.de/~kris

What is this course about?

This course

- introduces the basic concepts of Prolog programming,
- gets you started on programming yourself, and
- shows you how to do some natural language processing using Prolog.

Overview

Day 1: Prolog as a system for specifying and querying knowledge bases

Day 2: how Prolog finds answers to queries

Day 3: an important data structure: *lists*

Day 4: specifying grammars in Prolog

Day 5: building parsers in Prolog

Organization

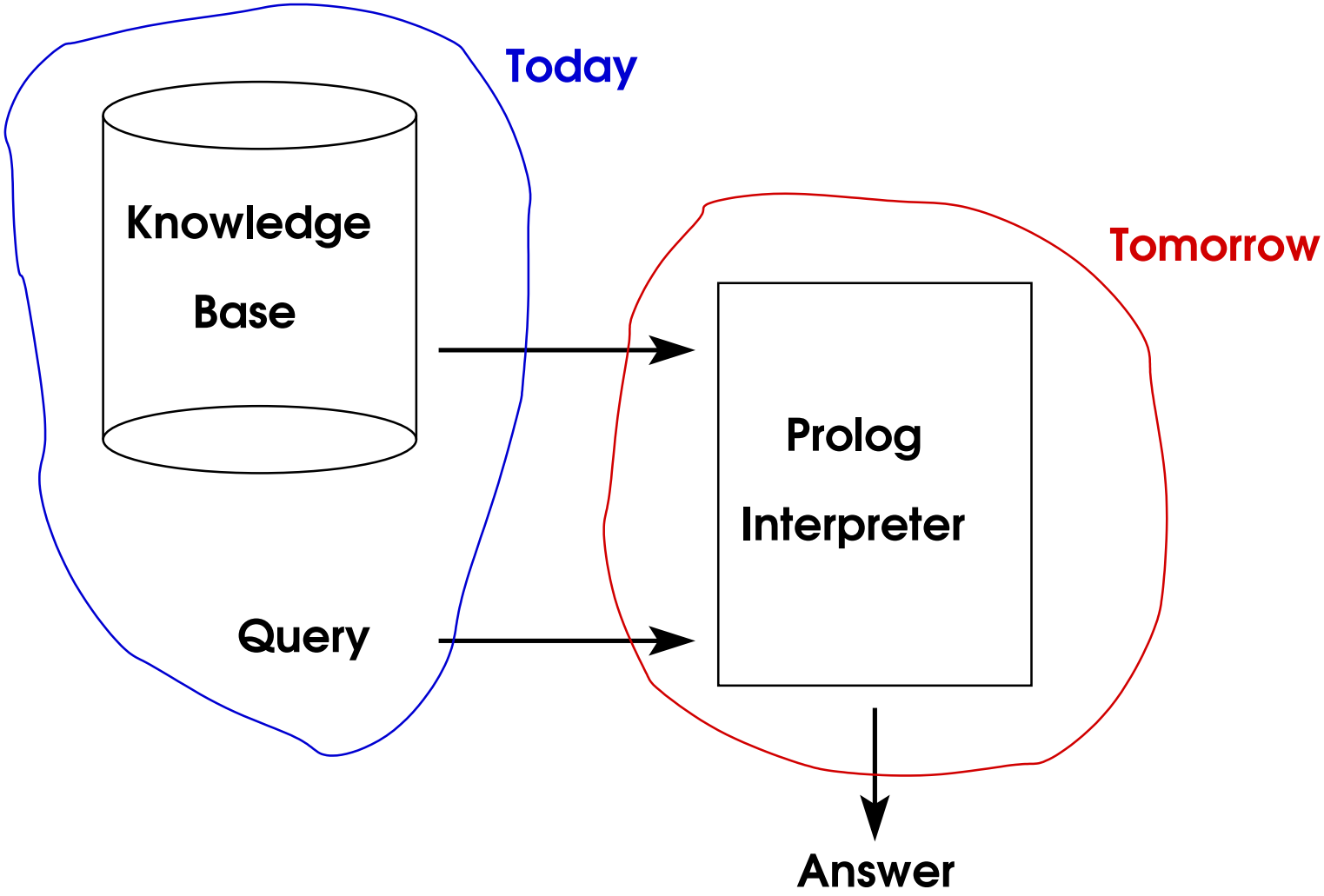
- Each session has a lecture part and a practical part (more about the practical part later).
- The reader contains the first eight chapters of *Learn Prolog Now!* by Blackburn, Bos, and Striegnitz. It also contains some questions (with answers) for each day, if you want to review what we have done.
- The slides are available at the course homepage `http://www.coli.uni-sb.de/~kris/esslli04prolog`.
- Questions during class: Ask any time and as many as possible.
- Questions outside class: You can find me near the ESSLLI desk from 10:30 to 11:30.

Day 1: Facts, Rules, and Queries

Today: How to specify knowledge bases in Prolog and how to query them.

Reader: Lecture 1 of *Learn Prolog Now!*

Prolog as a system for querying knowledge bases



kb1: A knowledge base of facts

```
wizard(harry).
wizard(ron).
wizard(hermione).

muggle(uncle_vernon).
muggle(aunt_petunia).

chases(crookshanks, scabbars).
```

kb1: queries we can ask

```
?- wizard(harry).
yes
?- chases(crookshanks,scabbars).
yes
?- muggle(harry).
no
?- muggle(dumbledore).
no
?- wizard(dumbledore).
no
?- witch(hermione).
ERROR: Undefined procedure:  witch/1
```

```
wizard(harry).
wizard(ron).
wizard(hermione).

muggle(uncle_vernon).
muggle(aunt_petunia).

chases(crookshanks,scabbars).
```

kb1: more queries we can ask

```
?- muggle(X).
X = uncle_vernon  ;
X = aunt_petunia  ;
no
?- chases(X,Y).
X = crookshanks
Y = scabbars  ;
no
?- chases(X,X).
no
```

```
wizard(harry).
wizard(ron).
wizard(hermione).

muggle(uncle_vernon).
muggle(aunt_petunia).

chases(crookshanks,scabbars).
```

A bit of syntax: atoms and variables

Atoms:

- All terms that consist of letters, numbers, and the underscore and start with a non-capital letter are **atoms**: `harry`, `uncle_vernon`, `ritaSkeeter`, `nimbus2000`, `....`
- All terms that are enclosed in single quotes are **atoms**: `'Professor Dumbledore'`, `'(@ *+ '`, `....`
- Certain special symbols are also atoms: `+`, `,`, `....`

Variables:

- All terms that consist of letters, numbers, and the underscore and start with a capital letter or an underscore are **variables**: `X`, `Hermione`, `_ron`, `....`
- _ is an anonymous variable: two occurrences of _ are different variables.

A bit of syntax: complex terms

Complex terms:

- **Complex terms** are of the form: *functor***(***argument*, ..., *argument***)**.
- Functors have to be atoms.
- Arguments can be any kind of Prolog term, e.g., complex terms.

  `likes(ron,hermione)`, `likes(harry,X)`,

  `f(a,b,g(h(a)),c)`, ...

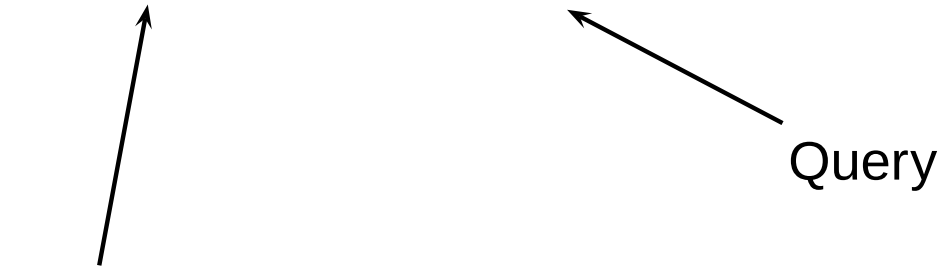
A bit of syntax: facts and queries

- **Facts** are complext terms which are followed by a full stop.

```
wizard(hermione).
muggle(uncle_vernon).
chases(crookshanks,scabbars).
```

- **Queries** are also complext terms which are followed by a full stop.

```
?- wizard(hermione).
```



Prompt provided by the Prolog interpreter.

kb2: a knowledge base of facts and rules

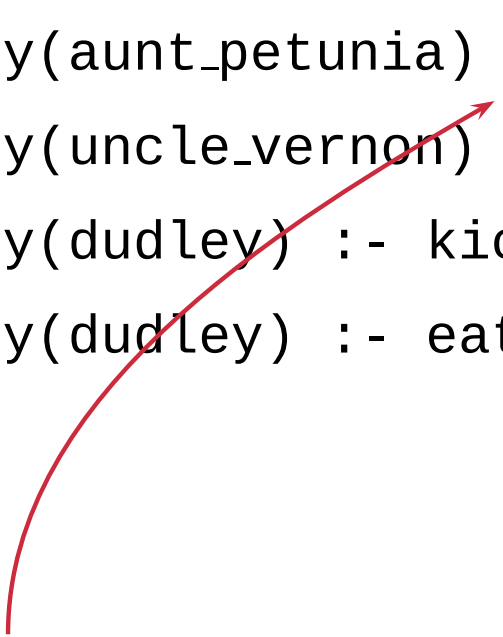
```
eating(dudley).

happy(aunt_petunia) :- happy(dudley).
happy(uncle_vernon) :- happy(dudley), unhappy(harry).
happy(dudley) :- kicking(dudley,harry).
happy(dudley) :- eating(dudley).
```

kb2: a knowledge base of facts and rules

```
eating(dudley).

happy(aunt_petunia) :- happy(dudley).
happy(uncle_vernon) :- happy(dudley), unhappy(harry).
happy(dudley) :- kicking(dudley,harry).
happy(dudley) :- eating(dudley).
```

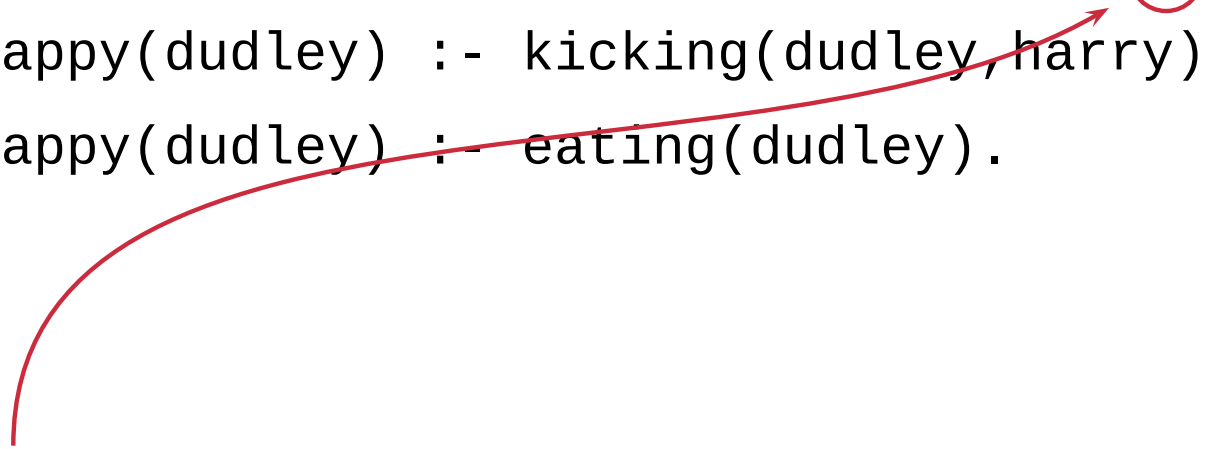


if ... then ...: If `happy(dudley)` is true, then `happy(aunt_petunia)` is true.

kb2: a knowledge base of facts and rules

```
eating(dudley).

happy(aunt_petunia) :- happy(dudley).
happy(uncle_vernon) :- happy(dudley), unhappy(harry).
happy(dudley) :- kicking(dudley,harry).
happy(dudley) :- eating(dudley).
```



and: If `happy(dudley)` is true and `unhappy(harry)` is true, then `happy(uncle_vernon)` is true.

kb2: a knowledge base of facts and rules

```
eating(dudley).

happy(aunt_petunia) :- happy(dudley).
happy(uncle_vernon) :- happy(dudley), unhappy(harry).
happy(dudley) :- kicking(dudley,harry).
happy(dudley) :- eating(dudley).
```

or: If `kicking(dudley,harry)` is true or if `eating(dudley)` is true, then `happy(dudley)` is true.

Querying kb2

```
?- happy(dudley).
yes
?- happy(aunt_petunia).
yes
?- happy(uncle_vernon).
no
?- happy(X).
X = aunt_petunia  ;
X = dudley  ;
no
```

```
eating(dudley).

happy(aunt_petunia) :- happy(dudley).

happy(uncle_vernon) :- happy(dudley),
                       unhappy(harry).

happy(dudley) :- kicking(dudley,harry).

happy(dudley) :- eating(dudley).
```

A bit of syntax: rules

- **Rules** are of the form *Head* `:-` *Body*.
- Like facts and queries, they have to be followed by a full stop.
- *Head* is a complex term.
- *Body* is complex term or a sequence of complex terms separated by commas.

```
happy(aunt_petunia) :- happy(dudley).
happy(uncle_vernon) :- happy(dudley),
                       unhappy(harry).
```

kb3: facts and rules containing variables

```
father(albert,james).
father(james,harry).
mother(ruth,james).
mother(lili,harry).

wizard(lili).
wizard(ruth).
wizard(albert).

wizard(X) :- father(Y,X),
             wizard(Y),
             mother(Z,X),
             wizard(Z).
```

This knowledge base defines 3 predicates: `father/2`, `mother/2`, and `wizard/1`.

For all `X`, `Y`, `Z`, if `father(Y,X)` is true and `wizard(Y)` is true and `mother(Z,X)` is true and `wizard(Z)` is true, then `wizard(X)` is true. I.e., for all `X`, if `X`'s father and mother are wizards, then `X` is a wizard.

Querying kb3

```
father(albert,james).
father(james,harry).
mother(ruth,james).
mother(lili,harry).

wizard(lili).
wizard(ruth).
wizard(albert).

wizard(X) :- father(Y,X),
             wizard(Y),
             mother(Z,X),
             wizard(Z).
```

```
?- wizard(james).
yes

?- wizard(harry).
yes

?- wizard(X).
X = lili ;
X = ruth ;
X = albert ;
X = james ;
X = harry ;
no

?- wizard(X), mother(Y,X), wizard(Y).
X = james
Y = ruth ;
X = harry
Y = lili ;
no
```

Prolog terms (overview)

atoms: Start with non-capital letters or are enclosed in single quotes.

`harry`, `nimbus2000`, `'Professor Dumbledore'`, `aunt_petunia`

numbers 3, 6, 2957, 8.34, ...

variables Start with a capital letter or an underscore.

`Harry`, `_harry`

complex terms An atom (the functor) is followed by a comma separated sequence of Prolog terms enclosed in parenthesis (the arguments).

`like(harry, X)`, `np(det(the),n(potion))`

Practical Session

- Go to the course homepage: `http://www.coli.uni-sb.de/~kris/esslli04prolog`.
- There you find links to the material for the practical sessions.
- Today's practical session starts with an explanation of how to use the Prolog interpreter.
- All exercises come with a page of hints and a solution. But don't peek before you have really tried to solve it yourself!
- You don't have to do all exercises. Decide for yourself whether you want to stick to the basic ones for the moment or whether you want to skip some of them to also have a look at the more advanced ones.
- If you need help, yell! If I am busy, ask your neighbors.