Künstliche Intelligenz

Einführung in Prolog
Teil 1

Dr. Claudia Schon
schon@uni-koblenz.de

Institute for Web Science and Technologies
Universität Koblenz-Landau

WeST
People and Knowledge Networks

UNIVERSITÄT KOBLENZ · LANDAU

Deklaratives Programmieren

- „Programmation en Logic“
- logische Programmiersprache
- deklaratives Programmieren

Deklaratives Programmieren

- Idee:
  - beschreibe die Situation, die von Interesse ist
  - stelle eine Frage
- Prolog:
  - leitet neue Fakten über die beschriebene Situation her
  - liefert so eine Antwort

Einführung in Prolog

Die folgenden Folien stammen aus dem ESSLI 2004 Kurs
„Prolog programming: a do-it-yourself course for beginners“
von Kristina Striegnitz:
http://cs.union.edu/~striegnk/courses/esslli04prolog/

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).
wizard(ron).
wizard(hermione).

muggle(uncle_vernon).
muggle(aunt_petunia).

chases(crookshanks,scabbars).
```

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

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).
```

Query

Prompt provided by the Prolog interpreter.

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).
```

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).

kicking(dudley,ron).
unhappy(ron).
```

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).

kicking(dudley,ron).
unhappy(ron).
```

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).

kicking(dudley,ron).
unhappy(ron).
```

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).

kicking(dudley,ron).
unhappy(ron).
```

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

Querying kb2

```
eating(dudley).

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

kicking(dudley,ron).
unhappy(ron).
```

```
?- happy(dudley).
?- happy(aunt_petunia).
?- happy(uncle_vernon).
?- happy(X).
```

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 defi nes 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

Take a look at the exercises in „Getting Started“ and „Basic Exercises“ which you can find here:

http://cs.union.edu/~striegnk/courses/esslli04prolog/practical.day1.php