Contents

Prolog 1

Basic idea 1

Consequences 2

Knowledge base 1 2

Knowledge base 2 2

Knowledge base 3 3

Expressing Disjunction 3

Prolog and Logic 3

Knowledge base 4 3

Knowledge base 5 4

Prolog Syntax 4

Atoms 4

Variables 5

Complex Terms 5

Examples 5

Arity 5

Functor 6

Prolog

- “Programming with logic”
- Declarative
- Very different from other programming language
- Good for knowledge rich tasks

Basic idea

- Describe the situation of interest
- Ask a question
- Prolog logically deduces new facts about the situation we described
- Prolog gives us its deducations back as answers

Consequences

- Think declaratively, not procedurallu
  - Challenging
  - Requires a different mindset
- High-level language
  - Not as efficient as, say, C
  - Good for rapid prototyping
  - Useful in many AI applications

Knowledge base 1

```
woman(mia).
woman(jody).
woman(yolanda).
playsAirGuiter(jody).
party.

?- woman(mia).
yes
?- playsAirGuitar(jody).
yes
?- playsAirGuitar(mia).
no
?- tattoed(jody).
ERROR: predicate tattoed/1 not defined
?- party.
yes
?- rockConcert.
ERROR: predicate rockConcert/1 not defined
```

Knowledge base 2

```
happy(yolanda).
listens2music(mia).
listens2music(yolanda):- happy(yolanda).
playsAirGuitar(mia):- listens2music(mia).
playsAurGuitar(yolanda):- listens2music(yolanda).

?- playsAirGuitar(mia).
yes
?- playsAirGuitar(yolanda).
yes
```

- Fact: `happy(yolanda)`
- Rule: `listens2music(yolanda):- happy(yolanda)`
- The end of a clause is marked with a full stop.
- Predicates: `happy`, `listens2music`, `playsAirGuitar`

Knowledge base 3

```
happy(vincent).
listens2music(butch).
playsAirGuitar(vincent):- listens2music(vincent), happy(vincent).
playsAirGuitar(butch):- happy(butch).
playsAirGuitar(butch):- listens2music(butch).

?- playsAirGuitar(butch).
yes
```

Expressing Disjunction

```
playsAirGuitar(butch):- happy(butch).
playsAirGuitar(butch):- listens2music(butch).
```

is the same as

```
playsAirGuitar(butch):- happy(butch); listens2music(butch).
```

Prolog and Logic

- Clearly Prolog has something to do with logic
- Operators
  - Implication: `:-`
  - Conjunection: `,`
  - Disjunction: `;`
- Use of modus ponens
- Negation

Knowledge base 4

```
woman(mia).
woman(jody).
woman(yolanda).
loves(vincent, mia).
loves(marsellus, mia).
loves(pumpkin, honey\_bunny).
loves(honey\_bunny, pumpkin).

?- woman(X).
X=mia;
X=jody;
X=yolanda;
no
?- loves(marsellus, X), woman(X).
X=mia
yes
?- loves(pumpkin, X), woman(X).
no
```

- Add ; to answer to get more possibilities

Knowledge base 5

```
loves(vincent, mia).
loves(marsellus, mia).
loves(pumpkin, honey\_bunny).
loves(honey\_bunny, pumpkin).
jealous(X,Y):- loves(X,Z), loves(Y,Z).

?- jealous(marsellus, W).
W=vincent
yes
```

Prolog Syntax

- What exactly are facts, rules and queries built out of?

Atoms

- A sequence of characters of upper-case letters, lower-case letters, digits, or underscore, starting with a lower case letter.
  - Exaples: butch, playGuitar
- An arbitrary sequence of characters enclosed in single quotes

  - 'Vincent', 'Five dollar shake'
- A sequence of special characters
  - ;, -

Variables

- A sequence of characters of upper-case letters, lower-case letters, digits, or underscore, starting with either an upper-case letter of an underscore

Complex Terms

- Atoms, numbers and variables are building blocks for complex terms
- Complex terms are built out of a functor directly followed by a sequence of arguments

Examples

- playsAirGuitar(jody).
- loves(vincent, mia).
- jealous(marsellus, W).
- hide(X, father(father(father(butch)))).

Arity

- The numbers of arguments a complex term has is called its *arity*
  - `woman(mia)` has arity 1
  - `loves(mia, vincent)` has arity 2
  - `hide(X, father(butch))` has arity 1
- In Prolog, you can define two predicates with the same functor but with different arity
- Prolog would treat this as two different predicates
- In Prolog, documentation arity of a predicate is usually indicated with the suffix "/" followed by a number to indicate the arity
  - `happy/1`
  - `listens2music/1`

Functor

- Things like the `playsAirGuitar` above. Functors must be atoms. The arity of a functor is the number of arguments it takes.
  - For example, `playsAirGuitar` has arity 1.
  - In documentation, the arity is usually noted as `playsAirGuitar/1`, meaning the functor has arity 1.
- Functors can be defined with the same name and different arities, though Prolog makes no assumption about whether those functors are related.