Introduction to AI with Prolog

Facts, Rules, and Queries

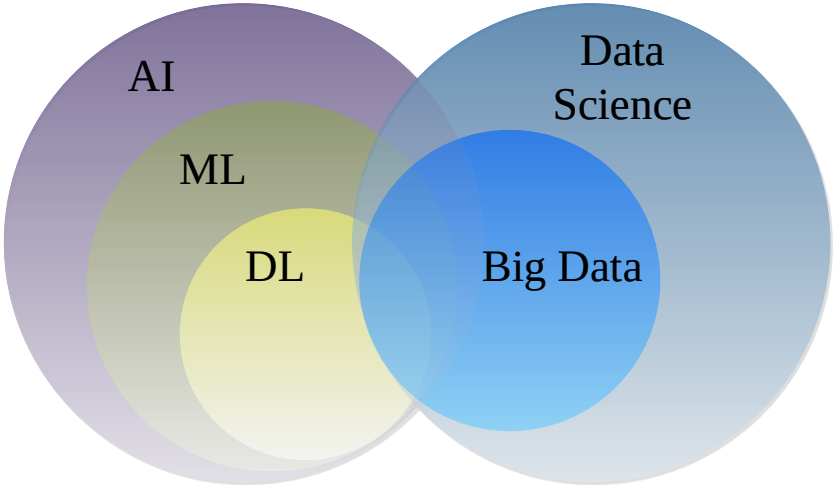
Alexey Sery
a.seryj@g.nsu.ru

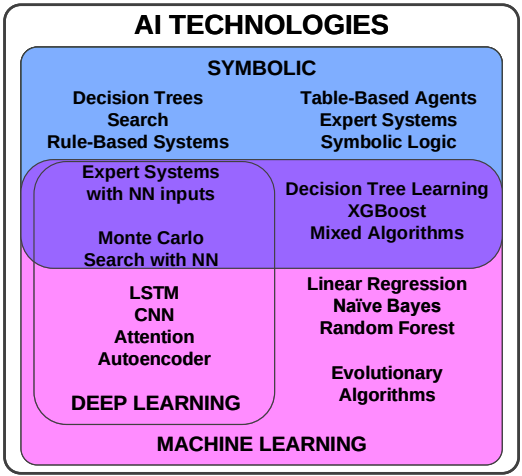
Novosibirsk State University
*THE REAL SCIENCE

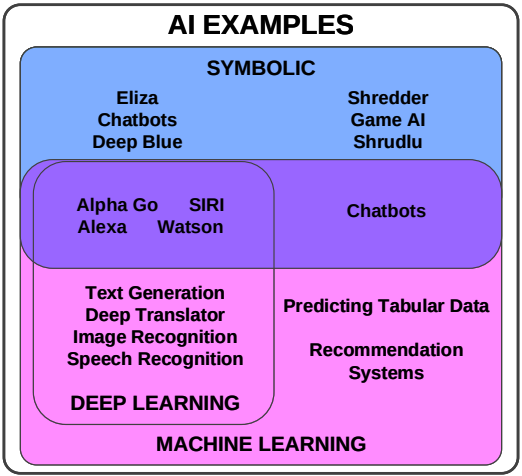
НГУ ФИТ

Department of Information Technologies
Novosibirsk State University

September 7, 2023


AI
ML
DL
Data
Science
Big Data


AI TECHNOLOGIES
SYMBOLIC
Decision Trees
Search
Rule-Based Systems
Table-Based Agents
Expert Systems
Symbolic Logic
Expert Systems
with NN inputs
Monte Carlo
Search with NN
Decision Tree Learning
XGBoost
Mixed Algorithms
LSTM
CNN
Attention
Autoencoder
DEEP LEARNING
Linear Regression
Naïve Bayes
Random Forest
Evolutionary
Algorithms
MACHINE LEARNING


AI EXAMPLES
SYMBOLIC
Eliza
Chatbots
Deep Blue
Shredder
Game AI
Shrudlu
Alpha Go
Alexa
SIRI
Watson
Chatbots
Text Generation
Deep Translator
Image Recognition
Speech Recognition
DEEP LEARNING
Predicting Tabular Data
Recommendation
Systems
MACHINE LEARNING

Logic programming is a variation of declarative programming based on a type of formal logic called *Predicate Calculus*. The precise algorithms and processing methods are left up to the language, which is expected to generate the proper outcome.

Logic programming should not be confused with programming logic, which is a more general study of how logical rules apply in computer programming.

Logic programs are completely data-driven and do not typically include any connective logic. Instead, the programs use a set of logical statements, which are also called *predicates*.

Logic programming allows for the concise representation of knowledge and the efficient execution of inference. It has been used in a wide range of AI applications, including natural language processing, planning, knowledge representation and reasoning.

Logic are declarative, meaning that they specify what is to be done, rather than how it is to be done. This makes them easier to understand and maintain than procedural programs.

Logic programs can be executed efficiently by computers. Logic programs can be easily extended and modified.

Logic programming is a well-understood paradigm with a rich theoretical foundation. This foundation can be used to develop new AI applications and to understand and improve existing ones.

Logic programming is well suited for use in distributed systems, such as the World Wide Web.

1. Logic programming is a type of AI that is based on formal logic. This means that it is based on a set of rules that are used to infer new information. It is a very powerful tool for AI, but it also has some limitations.
2. One of the biggest challenges with logic programming is that it can be very difficult to scale. This is because the number of rules that need to be considered grows exponentially as the size of the problem increases. This can make it very difficult to solve very large problems.
3. It can be difficult to deal with uncertain information. This is because the rules that are used to infer new information are based on a set of assumptions that may not be true in all cases. This can lead to incorrect results if the assumptions are not valid.
4. These challenges can be overcome with careful planning and design, but they need to be kept in mind when using logic programming for AI.

Tools and resources

We are going to use SWI-Prolog — a comprehensive free Prolog environment.

- You can download SWI-Prolog at this page:
  https://www.swi-prolog.org/download/stable
- Complete manual is available at
  https://www.swi-prolog.org/pldoc/doc_for?object=manual
- Online tool SWISH: https://swish.swi-prolog.org/
- Sources are available on github: https://github.com/Inscriptor/IntroductionToAI

Work and grading policy

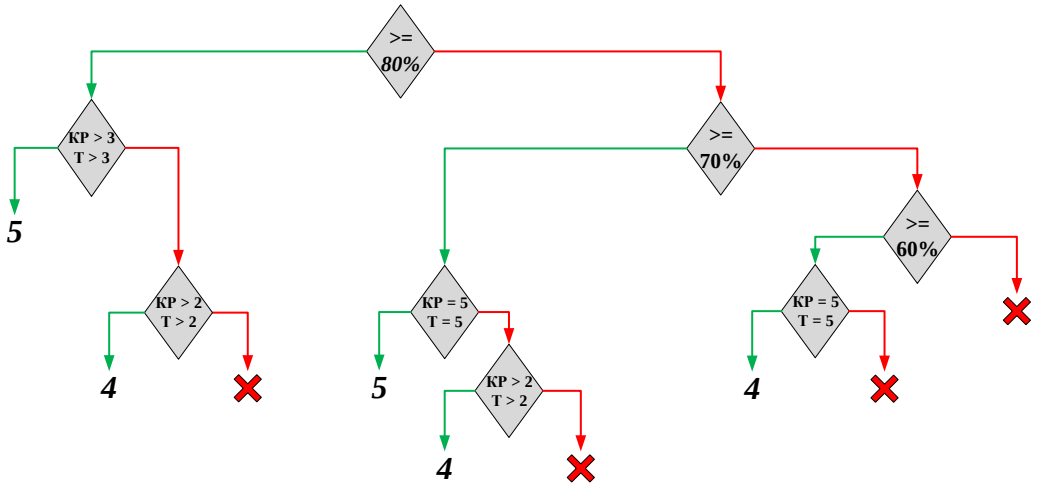
1 point = 1%.

Participation:	10%
Exercises:	10%
Assignments	5 assignments: 8% + 10% + 12% + 30% + 15% = 75%
Extra homeworks:	3 – 6% each
Class work:	0.2% for +

Work and grading policy

Late day policy:

- 7 free late days, afterwards 10% off per day late
- assignments not accepted after 1 week late per assignment
- homeworks are only accepted in classes next to those they were set in


>=
80%
KP > 3
T > 3
5
KP > 2
T > 2
4
>=
70%
KP = 5
T = 5
5
KP > 2
T > 2
4
>=
60%
KP = 5
T = 5
4

Overview

Goals of the lesson

- Look into the syntax of Prolog language
- Define the main terms of Prolog: fact, rule, query, and knowledge base
- Define basic syntax units, such as atom, variables and term
- Look at some example programs and try to figure out what they do
- Get out feet wet with some programming

- **Facts**. Facts are used to state things that are unconditionally true in the domain of interest.
- **Rules**. Rules state information that is conditionally true of the domain of interest.

Knowledge Base (aka database) is a collection of facts and rules. Prolog programs are knowledge bases, collections of facts and rules which describe some collection of relationships that we find interesting. We use knowledge base by posing queries.

Queries. The Prolog interpreter responds to queries about the facts and rules represented in its database. In making a query you are asking Prolog whether it can prove that your query is true. If so, it answers *True* and displays any *variable bindings* that it made in coming up with the answer. If it fails to prove the query true, it answers *False*.

Example 1: Simple knowledge base

Knowledge base

```
kind(fruit, orange).
kind(fruit, apple).
kind(veg, tomato).
kind(veg, onion).
```

Queries

Example 1: Simple knowledge base

Knowledge base

```
kind(fruit, orange).
kind(fruit, apple).
kind(veg, tomato).
kind(veg, onion).
```

Queries

```
?- kind(fruit, apple).  true.
?- kind(veg, apple).  false.
?- pred(veg, onion).  ERROR
```

Example 1: Simple knowledge base

Knowledge base

```
kind(fruit, orange).
kind(fruit, apple).
kind(veg, tomato).
kind(veg, onion).
color(red, apple).
color(red, tomato).
color(red, onion).
color(orng, orange)
```

Queries

```
?- kind(fruit, apple).  true.
?- kind(veg, apple).  false.
?- pred(veg, onion).  ERROR
```

Example 1: Simple knowledge base

Knowledge base

```
kind(fruit, orange).
kind(fruit, apple).
kind(veg, tomato).
kind(veg, onion).
color(red, apple).
color(red, tomato).
color(red, onion).
color(orng, orange)
```

Queries

```
?- kind(fruit, apple).  true.
?- kind(veg, apple).  false.
?- pred(veg, onion).  ERROR
?- color(red, F). F = apple; F =
tomato; F = onion.
?- kind(fruit, F), color(red, F).
F = apple.
```

Example 2: Conjunctions and disjunctions in rules

Knowledge base

```
lovesMusic(anna).
lovesMusic(sam).
playsInstrument(anna).
musician(P) :- lovesMusic(P),
playsInstrument(P).
melomaniac(P) :- !,lovesMusic(P);
playsInstrument(P).
```

Queries

Example 2: Conjunctions and disjunctions in rules

Knowledge base

```
lovesMusic(anna).
lovesMusic(sam).
playsInstrument(anna).
musician(P) :- lovesMusic(P),
playsInstrument(P).
melomaniac(P) :- !,lovesMusic(P);
playsInstrument(P).
```

Queries

```
?- musician(anna).
true.
?- musician(sam).
false.
?- melomaniac(anna).
true.
?- melomaniac(sam).
true.
?- melomaniac(Person).
Person=sam; Person=anna.
```

Example 3: Negations

Knowledge base

```
cat(fluffy).
cat(cornie).
bird(butch).
dog(bayley).
good(fluffy).
hasClaws(X) :- cat(X).
hasClaws(X) :- bird(X).
hasClaws(X) :- not(dog(X)).
animal(X) :- cat(X);bird(X);dog(X).
domestic(X) :-
animal(X),not(hasClaws(X)).
domestic(X) :- animal(X),good(X).
```

Queries

Example 3: Negations

Knowledge base

```
cat(fluffy).
cat(cornie).
bird(butch).
dog(bayley).
good(fluffy).
hasClaws(X) :- cat(X).
hasClaws(X) :- bird(X).
hasClaws(X) :- not(dog(X)).
animal(X) :- cat(X);bird(X);dog(X).
domestic(X) :-
animal(X),not(hasClaws(X)).
domestic(X) :- animal(X),good(X).
```

Queries

```
?- domestic(bayley).
true.
?- domestic(cornie).
false.
?- cat(C),domestic(C)
C = fluffy.
?- cat(C),not(domestic(C))
C = cornie.
```

Facts, rules and queries in Prolog are built of terms

- Atoms
- Numbers
- Strings
- Variables
- Compound terms — structures

Atoms

1. A string of characters made up of upper-case letters, lower-case letters, digits, and underscore characters, that begins with a lower-case letter.
2. An arbitrary sequence of characters enclosed in single quotes.
3. A sequence of special characters.

Example

```
chuck_norris, bayley, cornie, someCamelCaseStringNumber1, 'Chuck
Norris', 'Some arbitrary string', '@!?@', =====>, :-, @>
```

Numbers

1. **Real numbers**, though not particularly important in typical applications, are supported in Prolog.

Example

2.71828, 82.19284, $\pi, e, \ldots$

2. **Integers** are useful for many tasks, such as counting the elements of a list.

Example

$-2, -1, 0, 1, 2, 3, \ldots$

Variables and strings

1. A variable is a sequence of upper-case letters, lower-case letters, digits and underscore character that starts either with an upper-case letter or with underscore.

Example

```
X, Y, Var, Chuck_Norris, _someVar, _1zx, _
```

2. String is an arbitrary sequence of characters enclosed in double quotes.

Example

"Some arbitrary string line"

Compound terms

- Compound terms are built out of **functor (predicate)** followed by a sequence of **arguments**.
- The arguments are put in parentheses and are separated by commas.
- The functor of a term **must** be an atom.
- Arguments can be any kind of term.
- The number of arguments that a compound term has is called its **arity**.

Compound terms

- Any **constant** is a term. Constants are atoms, numbers or strings.
- Any **variable** is a term.
- Any sequence of a form `f(a1, a2, ...)` where `f` is an atom, and `a1`, `a2`, ... are terms, is a term.
- Conjunction and disjunction of terms are terms: `(T1, T2)`, `(T1; T2)`.
- There are no other terms.

Example

`g(f(x, y))`

Which of the character sequences are atoms, variables or neither of them?

1. `vARIABLE`
2. `Variable`
3. `x`
4. `XY1`
5. `chuck_norris_tells_simon_what_to_do`
6. `_john`
7. `'_jonh'`
8. `'John likes everybody'`
9. `Chuck Norris plays russian roulette with a fully loded revolver and wins`

Prolog Syntax

Which of the sequences are terms, and which are not. For every term indicate its functor and arity.

1. `loves(vincent,mia)`
2. `'loves(vincent,mia)'`
3. `Eats(cat,mouse)`
4. `hasChildren(cat,kittens)`
5. `and(musician(jody),artist(mia))`
6. `and(musician(X),artist(Y))`
7. `_and(musician(jody),artist(mia))`
8. `(Butch kills Vincent)`
9. `kills(Butch,Vincent)`

How many facts, rules, clauses and predicates there are in the knowledge base?

```
cat(fluffy).
cat(cornie).
bird(butch).
dog(bayley).
good(fluffy).
hasClaws(X) :- cat(X).
hasClaws(X) :- bird(X).
hasClaws(X) :- not(dog(X)).
animal(X) :- cat(X);bird(X);dog(X).
domestic(X) :- animal(X),not(hasClaws(X)).
domestic(X) :- animal(X),good(X).
```