Logical Languages
part 2

2020

Instructor: Odelia Schwartz

Formal logic and intro to predicate calculus

- All propositions can be expressed in **clausal form**

$$B_1 \cup B_2 \cup \ldots \cup B_n \subset A_1 \cap A_2 \cap \ldots \cap A_m$$

Right side implies left side
If all of the A are true, at least one B is true

Predicate calculus and proving theorems

- One way to simplify resolution process: restrict to simpler forms of propositions

Horn Clause

Either
(1) single atomic proposition on left side
(2) empty left side

Also called
(1) Headed horn clause
(2) Headless Horn clause

Predicate calculus and proving theorems

- One way to simplify resolution process: restrict to simpler forms of propositions

Horn Clause example:

(1) Headed horn clause

$$\text{likes}(\text{bob}, \text{trout}) \subset \text{likes}(\text{bob}, \text{fish}) \cap \text{fish}(\text{trout})$$

(2) Headless Horn clause

$$\text{father}(\text{bob}, \text{jake})$$

Often used to state fact

Most, but not all, propositions can be stated as Horn clauses. The restriction to Horn clauses makes resolution a practical process for proving theorems.

Predicate calculus and proving theorems

- Main idea: Presence of variables in propositions requires resolution to find values for the variables that allows matching to succeed

  - **Unification:** Finding values for variables in propositions that allows matching to succeed

  - **Instantiation:** temporary assigning of values to variables to allow unification

  - **Backtracking:** if resolution process to instantiate a variable with a value fails to complete required matching, then we backtrack and instantiate variable with different value

Prolog

We will use:

http://www.swi-prolog.org/

https://www.swi-prolog.org/pldoc/doc_for?object=manual

Prolog

You can run Prolog programs in several ways:
If you want to install it in your own machine, please install the latest version of the SWI-prolog. In this case please download it from http://www.swi-prolog.org/

Our lab also has the SWI-prolog. You can use it from there. To use it, first you need to login into the lab machines. Now you are ready to work with Prolog.

Prolog

Lets create a sample Prolog program file name simple.pl which has following two lines
person(bob).
father(bob,sam).
Now to open the Prolog complier write command swipl.
Now to load the simple.pl file type ['simple.pl']. in the compiler terminal. The ".” at the end of ['simple.pl']. is used to mark the end of the command in Prolog.
Then you can query based on this simple file: person(bob).
returns true
father(bob,X).
returns X = sam.
There are other instructions available:
http://www.cs.toronto.edu/~sheila/324/f05/tuts/swi.pdf
Ctrl-d to quit

Prolog

- University of Aix-Marselle (NLP) and Edinburgh (automated theorem proving) in mid 1970s
- Prolog dialect has several forms. Here we focus on Edinburgh syntax

Prolog

- Terms: 1. constant, 2. variable, 3. structure

Prolog

- Terms: 1. constant
- A constant is an atom or integer

Prolog

- Terms: 1. constant
- A constant is an atom or integer
- Atom: symbolic value of Prolog (similar to counterpart in LISP)

Prolog

- Terms: 1. constant
- A constant is an atom or integer
- Atom: symbolic value of Prolog (similar to counterpart in LISP)

➢ string of letters, digits, underscores beginning with lower case letter
Examples?

Prolog

- Terms: 1. constant
- A constant is an atom or integer
- Atom: symbolic value of Prolog (similar to counterpart in LISP)

➢ string of letters, digits, underscores beginning with lower case letter
Examples? likes, father, my_classes

Prolog

- Terms: 1. constant
- A constant is an atom or integer
- Atom: symbolic value of Prolog (similar to counterpart in LISP)

➢ String of letters, digits, underscores beginning with lower case letter
Examples? likes, father, my_classes
➢ String of printable ASCII characters
Examples? , :- have predefined meanings

Prolog

- Terms: 2. variable
- String of letters, digits, and underscores, beginning with Uppercase letters

Prolog

- Terms: 2. variable
- String of letters, digits, and underscores, beginning with Uppercase letters

  Examples: X, List

Prolog

- Terms: 2. variable
- String of letters, digits, and underscores, beginning with Uppercase letters
- Variables are not bound to types by declaration

Prolog

- Terms: 2. variable
- **Variable instantiation**: Binding a variable to a value and thus to a type

Prolog

- Terms: 2. variable

- **Variable instantiation**: Binding a variable to a value and thus to a type

➢ Lasts only as long as it takes to satisfy one complete goal (proof or disproof of proposition)

Prolog

- Terms: 2. variable

- **Variable instantiation**: Binding a variable to a value and thus to a type

➢ Lasts only as long as it takes to satisfy one complete goal (proof or disproof of proposition)

➢ Example: student(X)
instantiation will set a variable X to bob and check the proposition student(bob)

Prolog

- Terms: 2. variable
- Prolog variables only **distant relatives** to imperative languages both in semantics and use

Prolog

- Terms: 3. structure
- Represents atomic propositions of predicate calculus

Prolog

- Terms: 3. structure
- Represents atomic propositions of predicate calculus

  functor(parameter_list)

Prolog

- Terms: 3. structure
- Represents atomic propositions of predicate calculus

  functor(parameter_list)

  Example?

Prolog

- Terms: 3. structure
- Represents atomic propositions of predicate calculus

  functor(parameter_list)

  Example? father(jon, shelley)

Prolog

- Terms: 3. structure
- Represents atomic propositions of predicate calculus

  functor(parameter_list)

  Example? father(jon, shelley)

  Used for?

Prolog

- Terms: 3. structure
- Represents atomic propositions of predicate calculus

  functor(parameter_list)

  Example? father(jon, shelley)

  ➢ used to specify facts in Prolog

Prolog

- Terms: 3. structure
- Represents atomic propositions of predicate calculus

  functor(parameter_list)

  Example? father(jon, shelley)

  used to specify facts in Prolog

  ➢ also a predicate when specifying a question (query)

Prolog

Fact statements

- Construct hypotheses or database of assumed information; statements from which new information can be inferred

Prolog

Fact statements

- Construct hypotheses or database of assumed information; statements from which new information can be inferred
- Remember: facts we have in database; then queries/goals asking about database

Prolog

Statement forms

1. Headless Horn clauses of predicate calculus

Examples??

Prolog

Statement forms

1. Headless Horn clauses of predicate calculus

Examples??

female(shelley)
male(bill)
father(bill, jake)

Prolog

Statement forms

- Headless Horn clauses of predicate calculus

Examples??

female(**s**helley)
male(**b**ill)
father(**b**ill, **j**ake)

Why are the first letters of each term lower case?

Prolog

Statement forms

1. Headless Horn clauses of predicate calculus

Examples??

female(**s**helley)
male(**b**ill)
father(**b**ill, **j**ake)

Why are the first letters of each term lower case?
Answer: these are not variables, but facts (or queries)

Prolog

Statement forms

2. Rule statements (these will correspond to headed horn clauses)

Prolog

Statement forms

2. Rule statements (these will correspond to headed horn clauses)

consequence :- expression

Here the expression implies consequence (**right side implies left side**)

Prolog

Statement forms

2. Rule statements (these will correspond to headed horn clauses)

consequence :- expression

Here the expression implies consequence (right side implies left side)

The expression can be a single term or **conjunction**
Example: female(shelley), child(shelley)

Prolog

Statement forms

2. Rule statements (these will correspond to headed horn clauses)

Example headed horn clauses:

ancestor(mary, shelley) :- mother(mary, shelley)

Prolog

Statement forms

2. Rule statements (these will correspond to headed horn clauses)

Example headed horn clauses:

ancestor(mary, shelley) :- mother(mary, shelley)

Reads: If mary is the mother of shelley,
then this implies that mary is an ancestor of shelley

Prolog

Statement forms

2. Rule statements (these will correspond to headed horn clauses)

Use of Variables in Prolog statements

Prolog

Statement forms

2. Rule statements (these will correspond to headed horn clauses)

Use of Variables in Prolog statements

parent(X,Y) :- mother(X,Y)

Meaning?

Prolog

Statement forms

2. Rule statements (these will correspond to headed horn clauses)

Use of Variables in Prolog statements

parent(X,Y) :- mother(X,Y)

Meaning? If there are instantiations of X, Y such that mother(X,Y) is true, then for those instantiations of X and Y, parent (X,Y) is true

Prolog

Statement forms

2. Rule statements (these will correspond to headed horn clauses)

Use of Variables in Prolog statements

parent(X,Y) :- mother(X,Y)

Meaning? If there are instantiations of X, Y such that mother(X,Y) is true, then for those instantiations of X and Y, parent (X,Y) is true

There could be several X,Y pairs in the database for which parent(X,Y) is true. jon, shelley mary,liz etc

Prolog

Statement forms

2. Rule statements (these will correspond to headed horn clauses)

Use of Variables in Prolog statements

parent(X,Y) :- mother(X,Y)

Meaning? If there are instantiations of X, Y such that mother(X,Y) is true, then for those instantiations of X and Y, parent (X,Y) is true

Use of variables allows to generalize meanings

Prolog

Statement forms

3. Goal statements (these will correspond to headless horn clauses, like the fact statements)

Prolog

Statement forms

3. Goal statements (these will correspond to headless horn clauses)

- So far: we have described statements as logical propositions, for facts and logical relationships between facts. These are the basis for theorem proving.

Prolog

Statement forms

3. Goal statements (these will correspond to headless horn clauses)

- So far: we have described statements as logical propositions, for facts and logical relationships between facts. These are the basis for theorem proving.
- The theorem: in the form of a proposition that we want to prove or disprove (called goals or queries)

Prolog

Statement forms

3. Goal statements (these will correspond to headless horn clauses)

- So far: we have described statements as logical propositions, for facts and logical relationships between facts. These are the basis for theorem proving.
- The theorem: in the form of a proposition that we want to prove or disprove (called goals or queries)
  Example: man(fred)

Prolog

Statement forms

3. Goal statements (these will correspond to headless horn clauses)

Example: man(fred)

The system will respond either:

true: proved goal and true under database of facts and relations

false: either goal was determined as false, or system was unable to prove it

Prolog

Statement forms

3. Goal statements (these will correspond to headless horn clauses)

Another example: father(X, mike)

Prolog

Statement forms

3. Goal statements (these will correspond to headless horn clauses)

Another example: father(**X**, mike)

Note that **X** is a variable (starts with capital letter)

Prolog

Statement forms

3. Goal statements (these will correspond to headless horn clauses)

Another example: father(**X**, mike)

Note that **X** is a variable (starts with capital letter)

When a variable is present, the system not only asserts validity, but **identifies instantiations of variable that make goal true**

Prolog

Prolog demos

simple.pl file includes:

```
% Simple example for testing
% swipl from command line
% Inside compiler:
% ['simple.pl'].
% person(bob).
% returns true
% father(bob,X).
% returns X = sam.
% control d to exit

person(bob).
father(bob,sam).
```

Prolog

Prolog demos

simple.pl let's try it in compiler:

- swipl from command line
- Inside compiler:
  ['simple.pl'].

Notice we always have a period after statement

Prolog

Prolog demos

simple.pl let's try it in compiler:

- swipl from command line
- Inside compiler:
  ['simple.pl'].
- person(bob).
  Returns?

Prolog

Prolog demos

simple.pl let's try it in compiler:

- swipl from command line
- Inside compiler:
  ['simple.pl'].
- person(bob).
  Returns true

Prolog

Prolog demos

simple.pl let's try it in compiler:

- swipl from command line
- Inside compiler:
  ['simple.pl'].
- father(bob,X).
  Returns?

Prolog

Prolog demos

simple.pl let's try it in compiler:

- swipl from command line
- Inside compiler:
  ['simple.pl'].
- father(bob,X).
  Returns? X = sam.

Prolog

Prolog demos

simplemore.pl let's add more facts to file:

```
person(bob).
father(bob,sam).
father(sam,liz).
```

Prolog

Prolog demos

simplemore.pl let's add more facts to file:

```
person(bob).
father(bob,sam).
father(bob,liz).
```

- father(bob,X).
  Returns?

  initially returns X = sam

Prolog

Prolog demos

simplemore.pl let's add more facts to file:

```
person(bob).
father(bob,sam).
father(bob,liz).
```

- father(bob,X).
  Returns?

  initially returns X = sam
  Type ; and will return next item here:
  X = liz

Prolog

Prolog demos

simplemore.pl let's add more facts to file:

```
person(bob).
father(bob,sam).
father(bob,liz).
```

- father(bob,X).
  Returns?

  returns X = sam ; X = liz

  So system will attempt (called unification) to find instantiations of X that results in true value for goal

Prolog

Prolog demos

simple2.pl

```
%http://faculty.otterbein.edu/psanderson/csc326/notes/PrologNotes.html

mother(iva, pete).
mother(iva, ed).
mother(iva, becky).
mother(kay, nancy).
mother(kay, bob).
mother(kay, diane).
mother(becky, katie).
husband(dwight, iva).
husband(robert, kay).
husband(pete, nancy).
```

Prolog

Prolog demos

simple2.pl

%http://faculty.otterbein.edu/psanderson/csc326/notes/PrologNotes.html

```
mother(iva, pete).
mother(iva, ed).
mother(iva, becky).
mother(kay, nancy).
mother(kay, bob).
mother(kay, diane).
mother(becky, katie).
husband(dwight, iva).
husband(robert, kay).
husband(pete, nancy).
```

Things to try:
mother(kay, nancy).
mother(kay, kay).
mother(kay, Who). press ;

Prolog

Prolog demos

simple2.pl

%http://faculty.otterbein.edu/psanderson/csc326/notes/PrologNotes.html

```
mother(iva, pete).
mother(iva, ed).
mother(iva, becky).
mother(kay, nancy).
mother(kay, bob).
mother(kay, diane).
mother(becky, katie).
husband(dwight, iva).
husband(robert, kay).
husband(pete, nancy).
```

Things to try:
mother(kay, Who). press ;

```
mother(kay,Who).
Who = nancy ;
Who = bob ;
Who = diane.
```

Note about form

- Goal and non goal statements (e.g., facts, rules) can have the same form

Note about form

- Goal and non goal statements (e.g., facts, rules) can have the same form
- So Prolog implementation must have means to differentiate goals and non goals

Prolog

Note about form

- Goal and non goal statements (e.g., facts, rules) can have the same form
- So Prolog implementation must have means to differentiate goals and non goals
- We separated by reading in facts file first

Prolog

Inferencing process of Prolog

- Prolog resolution is critical (proving true, or false cannot prove)

Prolog

Inferencing process of Prolog

- Prolog resolution is critical (proving true, or false cannot prove)
- Queries are called goals
  If a goal is a compound proposition,
  it consists of subgoals
- To prove goal true:
  inferencing process must find chain of rules and/or facts in the database

Prolog

Inferencing process of Prolog

If Q is a goal, then either Q must be found in the database, or inferencing must find fact P1 and propositions P2, P3, P4, … Pn such that:

```
P2 :- P1
P3 :- P2
P4 :- P3

   …

Q  :- Pn
```

Prolog

Inferencing process of Prolog

If Q is a goal, then either Q must be found in the database, or inferencing must find fact P1 and propositions P2, P3, P4, … Pn such that:

```
P2 :- P1
P3 :- P2
P4 :- P3

   …

Q  :- Pn
```

Process is called matching, satisfying, or resolution

Prolog

Inferencing process of Prolog. Example:

man(bob) query

Prolog

Inferencing process of Prolog. Example:

man(bob) query

Easy if database includes this fact; then proof trivial

Prolog

Inferencing process of Prolog. Example:

man(bob) query

More complex if database includes rules:

```
father(bob).
man(X) :- father(X).
```

Prolog

Inferencing process of Prolog. Example:

man(bob) query

More complex if database includes rules:

father(bob).
man(X) :- father(X).

Prolog needs to find the two statements, and use them to infer the truth of the goal; needs unification to instantiate X temporarily to bob

Prolog

Inferencing process of Prolog. Example:

man(X) query

More complex if database includes rules:

```
father(bob).
father(jon).
man(X) :- father(X).
```

Prolog must match goal against propositions in database

e.g., first find bob then jon (remember we used ; in compiler)

Prolog

Inferencing process of Prolog. Example:

man(bob) query

Database includes rules:
father(bob).
man(X) :- father(X).

How does Prolog do it? Two possibilities:

1. Forward chaining: search for and find first proposition father(bob); goal is inferred by matching first proposition with right side of second rule father(X) through instantiation of X to bob, and then matching left side of second proposition to goal man(bob)

Prolog

Inferencing process of Prolog. Example:

man(bob) query

Database includes rules:
father(bob).
man(X) :- father(X).

How does Prolog do it? Two possibilities:

2. Backward chaining: first match goal with left side of second proposition man(X) through the instantiation of X to bob; as last step, match right side of second proposition (now father(bob)) with first proposition

Prolog

How does Prolog do it? Two possibilities:

1. Forward chaining: search for and find first proposition father(bob); goal is inferred by matching first proposition with right side of second rule father(X) through instantiation of X to bob, and then matching left side of second proposition to goal man(bob)

2. Backward chaining: first match goal with left side of second proposition man(X) through the instantiation of X to bob; as last step, match right side of second proposition (now father(bob)) with first proposition

Which does Prolog use?

Prolog

How does Prolog do it? Two possibilities:

1. Forward chaining: search for and find first proposition father(bob); goal is inferred by matching first proposition with right side of second rule father(X) through instantiation of X to bob, and then matching left side of second proposition to goal man(bob)

2. Backward chaining: first match goal with left side of second proposition man(X) through the instantiation of X to bob; as last step, match right side of second proposition (now father(bob)) with first proposition

Prolog uses Backward chaining. First match goal.