UNICAM
Università di Camerino
1336

Logic and Constraint Programming

PROLOG

Prof. Fabrizio Fornari

May 6, 2022

About me

Fabrizio Fornari

Research fellow @ Unicam Computer Science Dpt.

fabrizio.fornari@unicam.it

https://pros.unicam.it/members

Some of my interests and research topics:

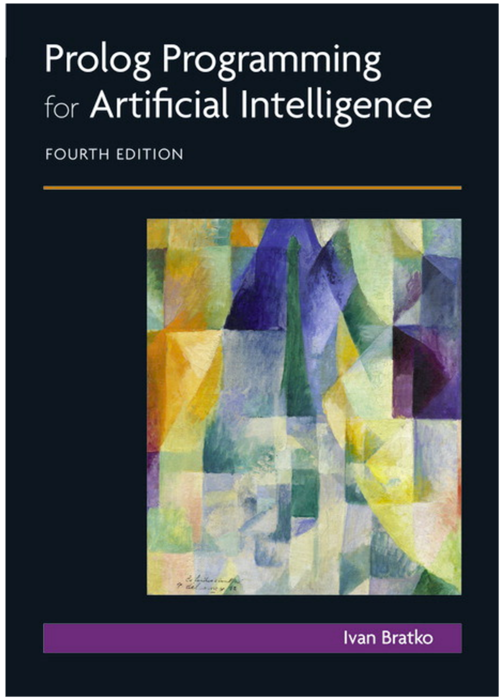
- Business Process Management
- Business Process Modelling and Verification
- Internet of Things
- Software Engineering
- Model-Driven Engineering for IoT
- Process Mining
- Digital Twin

Logic Programming

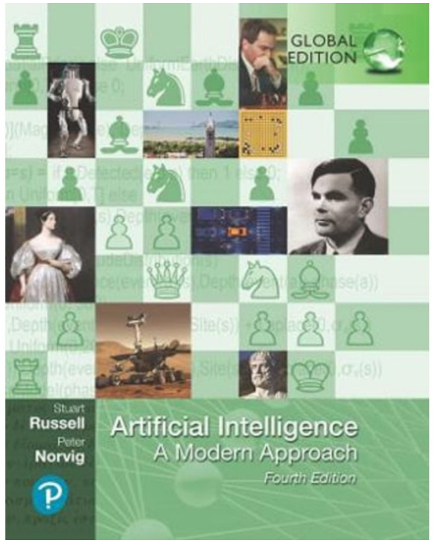
Prolog is a logic programming language associated with artificial intelligence and computational linguistics

SWI-Prolog is a versatile implementation of the Prolog language.

Support Material



- Bratko, Ivan. *Prolog programming for artificial intelligence*. 4th edition Pearson education, 2011.



- Stuart J. Russell and Peter Norvig. *Artificial Intelligence A Modern Approach*. Fourth Edition. Pearson, 2020.

Evaluation

During the course we assign **4 practical exercises** to solve outside course hours using the **tools** introduced during the course.

Assignments are **mandatory** for the final examination. They must be delivered **5 days before** the exam.

The **exam** consists of a **discussion** of the **assignments** and answers to **questions** on the topic treated during the course.

Exam Dates

- 30/06/2022 - Last day to deliver the assignments 24/06/2022 (midnight)
- 14/07/2022 - Last day to deliver the assignments 08/07/2022 (midnight)
- 28/07/2022 - Last day to deliver the assignments 22/07/2022 (midnight)

- LCP Wiki Page

  http://didattica.cs.unicam.it/doku.php?id=didattica:ay2122:lcp:main

Any Question?

Preamble

- Declarative vs Imperative programming

Do you know the difference?

Different level of abstraction

Declarative vs Imperative programming

Declarative

- "what to do, not how to do it”

Imperative

- “how to do it, not what to do ”

Which one refers to a higher level of abstraction?

Declarative vs Imperative programming

Declarative

- "what to do, not how to do it”
- Higher level of abstraction

Imperative

- “how to do it, not what to do ”
- Lower level of abstraction

Which one is the best?

Declarative vs Imperative programming

Declarative

- "what to do, not how to do it”
- Higher level of abstraction

Imperative

- “how to do it, not what to do ”
- Lower level of abstraction

Neither one of them is better or worse,
but both have their places

Declarative vs Imperative programming

Declarative

- "what to do, not how to do it”
- Higher level of abstraction
- Ex. database query languages, such as SQL

Imperative

- “how to do it, not what to do ”
- Lower level of abstraction
- Ex. Java, C, C++, Python..

Imperative Programming provides flexibility but brings in complexity
Declarative programming hides complexity and provides simplicity

In practice, **mixed forms of the paradigms** are generally used, have their advantages and disadvantages.

What about PROLOG?

Is it Declarative or Imperative..?

PROLOG

PROgramming in LOGic (PROLOG) is a **declarative programming language**.

In Prolog, **we do not write out what the computer should do line by line**, as in *imperative* languages such as C and Java .

In prolog we describe a situation. Based on this code, **the interpreter or compiler will tell us a solution**.

The computer will tell us whether a prolog sentence is true or not and, if it contains variables, what the values of the variables need to be.

PROLOG

Prolog is most useful in the areas related to artificial intelligence research, such as problem solving, (path) planning or natural language interpretation.

First Appeared: 1972; (java in 1995; python in 1991, C in 1972)

Colmerauer, A., & Roussel, P. (1996). **The birth of Prolog**. In *History of programming languages---II* (pp. 331-367).

First Order Logic

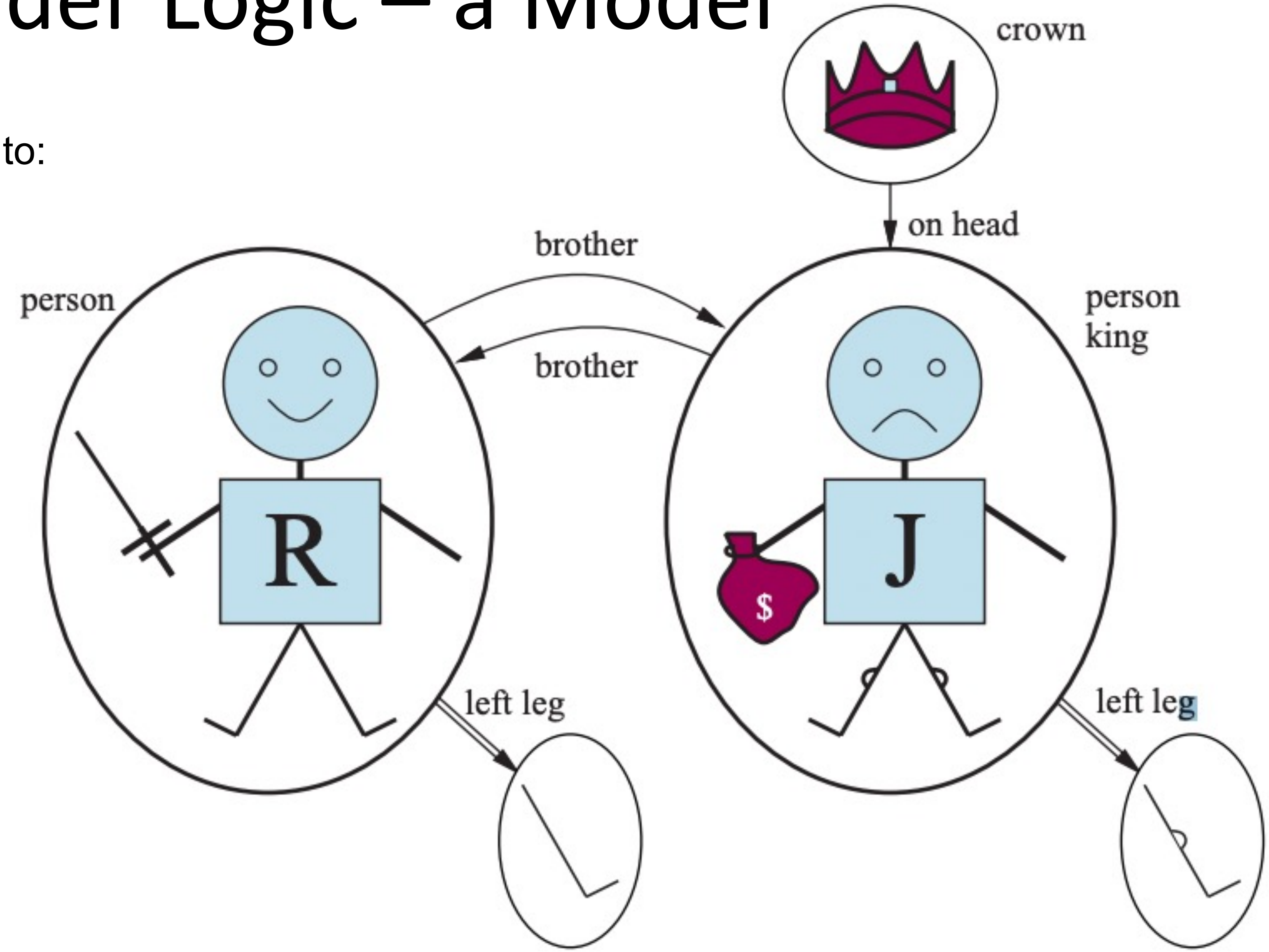
Prolog has its roots in first order logic (also known as predicate logic)

- Objects (cat, dog, house, Bob, etc.)
- Relations (has color, bigger than, mother of, father of, etc.)
- Facts: (One value for a given input: has father, has head, can swim, etc.)

Facts have a truth value. *True* or *False*

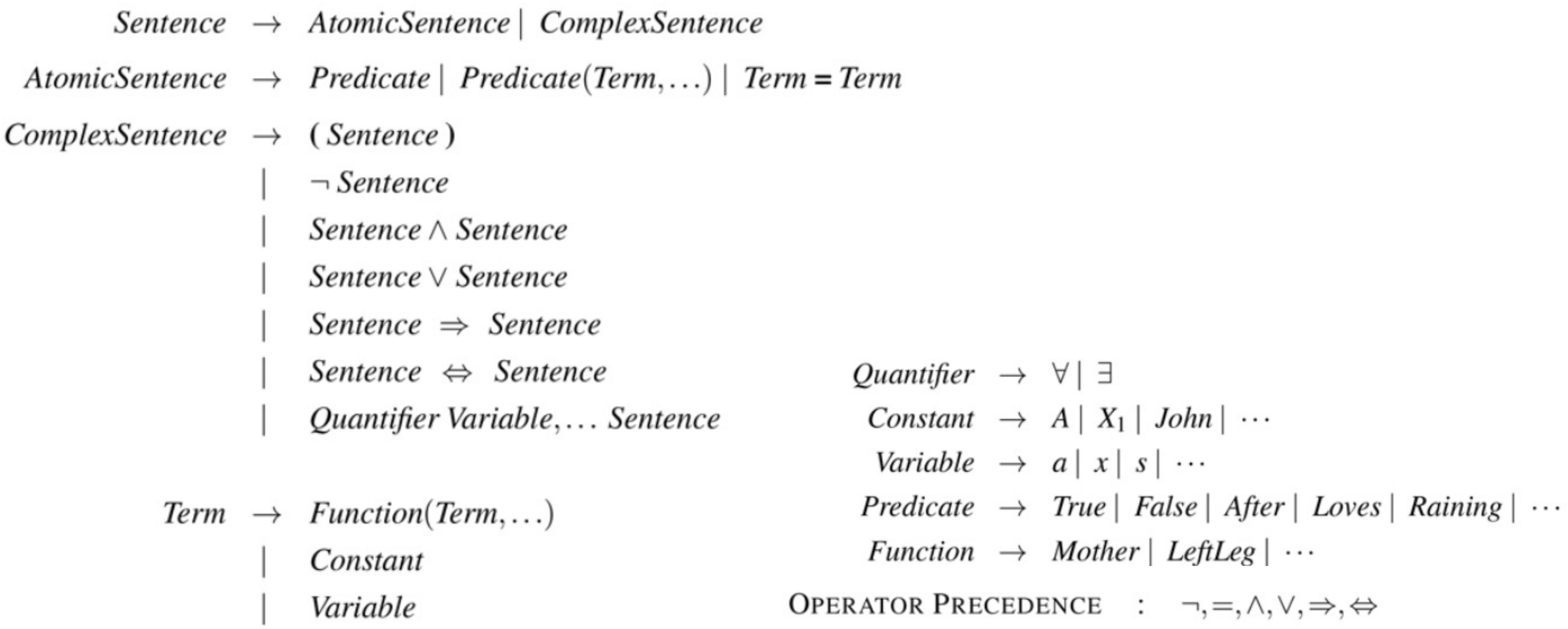
First Order Logic – a Model

Dissect the world into:
- Objects
- Relations
- Facts



Russell, S., & Norvig, P. Artificial intelligence: a modern approach. 4th Ed. (2020).

First Order Logic - Syntax



$$
\begin{array}{rcl}
\mathit{Sentence} & \rightarrow & \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{AtomicSentence} & \rightarrow & \mathit{Predicate} \mid \mathit{Predicate}(\mathit{Term}, \ldots) \mid \mathit{Term} = \mathit{Term} \\
\mathit{ComplexSentence} & \rightarrow & (\,\mathit{Sentence}\,) \\
 & \mid & \neg\, \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \land \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \lor \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \;\Rightarrow\; \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \;\Leftrightarrow\; \mathit{Sentence} \\
 & \mid & \mathit{Quantifier}\ \mathit{Variable}, \ldots\ \mathit{Sentence} \\
 & & \\
\mathit{Term} & \rightarrow & \mathit{Function}(\mathit{Term}, \ldots) \\
 & \mid & \mathit{Constant} \\
 & \mid & \mathit{Variable}
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{Quantifier} & \rightarrow & \forall \mid \exists \\
\mathit{Constant} & \rightarrow & A \mid X_1 \mid \mathit{John} \mid \cdots \\
\mathit{Variable} & \rightarrow & a \mid x \mid s \mid \cdots \\
\mathit{Predicate} & \rightarrow & \mathit{True} \mid \mathit{False} \mid \mathit{After} \mid \mathit{Loves} \mid \mathit{Raining} \mid \cdots \\
\mathit{Function} & \rightarrow & \mathit{Mother} \mid \mathit{LeftLeg} \mid \cdots
\end{array}
$$

$$
\textsc{Operator Precedence} \quad : \quad \neg, =, \land, \lor, \Rightarrow, \Leftrightarrow
$$

Russell, S., & Norvig, P. Artificial intelligence: a modern approach. 4th Ed. (2020).

First Order Logic - Syntax

Facts, Predicates, or Atomic sentences:

P(x,y) is read as “ x is P of y.”   argument-ordering convention

Brother(Richard, John)

First Order Logic - Syntax

Facts, Predicates, or Atomic sentences:

Brother(Richard, John) (predicate is true, if its first element is Richard and the second is John*)*

HasWheels(Car) (predicate is true, if its first element is car)

MotherOf(Charles, Elizabeth) (predicate is true, if its first element is Charles and the second is Elizabeth)

Usually there is a collection of sentences that are assumed to be true, to create a logical definition of predicates.

Such a collection of sentences that are true is called a **knowledge base**.

First Order Logic – Sentences

Predicates are relations while Functions will return a value

Fact, Predicates, Atomic sentence, ex. *Brother(Richard, John)*

Fucntions ex. Bro(John) = Richard

Complex Sentence ex. Brother(R,J) Λ Brother (J,R)

¬*King(Richard) => King(John)*

Universal Quantifiers: ∀ King(x) => Person(x)

Existential Quantifiers: ∃ Crown(x) Λ onHead(x, John)

First Order Logic - Sentences

Other sentences:

King(Richard) \/ King(John)

∀*x* ∀*y Brother(x,y) => Sibling(x,y)*

In(Paris,France) /\ In(Marseilles, France)

∀ c Country(c) /\ Border(c, Ecuador) => In(c, SouthAmerica)

∃ Country(c) /\ Border(c, Spain) /\ Border(c, Italy)

From English to First Order Logic

- Richard has only two brothers, John and Geoffrey:

$$Brother(John, Richard) \land Brother(Geoffrey, Richard) \land John \neq Geoffrey \land$$
$$\forall x Brother(x, Richard) => (x = John \lor x = Geoffrey)$$

- No Region in South America borders any region in Europe

$$\forall c, d\ In(c, SouthAmerica) \land In(d, Europe) => \neg Border(c,d)$$

- No two adjacent countries have the same map color

$$\forall x,y\ Country(x) \land Country(y) \land Border(x,y) =>$$
$$\neg (Color(x) = Color(y)) \land \neg (x = y)$$

Logic in Prolog

The **logic used in prolog is a version of first order logic**, with the use of capital letters inverted (predicates and objects start with a lowercase letter, variables start with an uppercase letter).

A prolog program consists of a knowledge base where each sentence is a conjunction of predicates connected to a final predicate with an implication.

For instance:

$$\forall a, b, c, d Pred1(a, b) \wedge Pred2(b, c) \wedge Pred3(c, d) \Rightarrow Pred4(a)$$

A sentence like this is called a Horn Clause.

Logic in Prolog

$$\forall a, b, c, d Pred1(a, b) \wedge Pred2(b, c) \wedge Pred3(c, d) \Rightarrow Pred4(a)$$

In prolog the above sentence would look like this:

```
pred4(A) :- pred1(A, B), pred2(B, C), pred3(C, D).
```

Note that the implication sign is reversed, commas are used for conjunction, a period is used to end the sentence and all variables are assumed to be universally quantified.

Prolog from Theory to Practice

SWI-Prolog

SWI-Prolog is a versatile implementation of the Prolog language. Although SWI-Prolog gained its popularity primarily in education, its development is mostly driven by the needs for **application development**.

SWI-Prolog aims at **scalability**. Its robust support for multi-threading exploits multi-core hardware efficiently and simplifies embedding in concurrent applications.

SWI-Prolog **unifies many extensions** of the core language that have been developed in the Prolog community such as *tabling*, *constraints*, *global variables*, *destructive assignment*, *delimited continuations* and *interactors*.

Let us download SWI Prolog

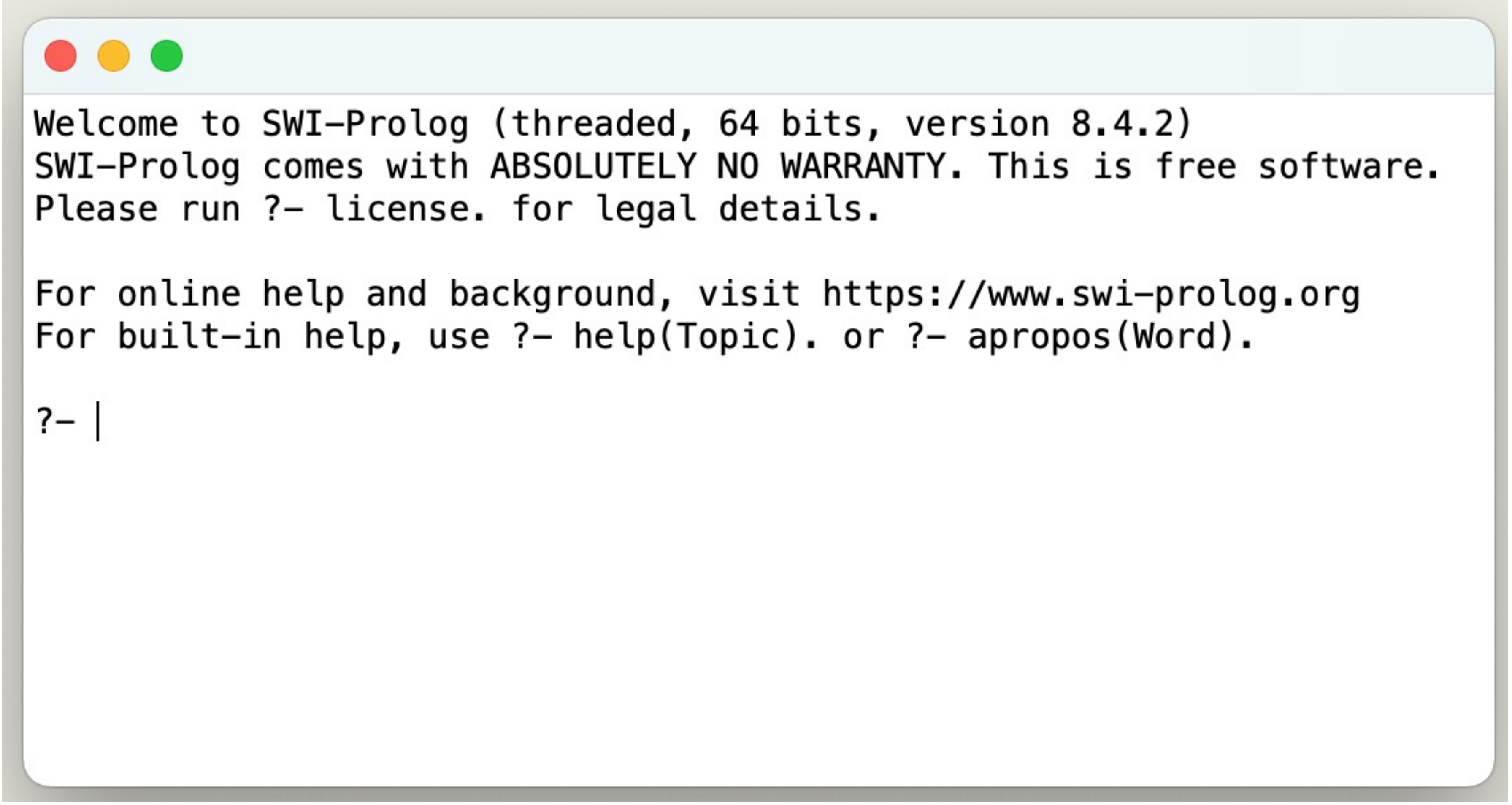
Stable version:

https://www.swi-prolog.org/download/stable

SWI Prolog documentation:

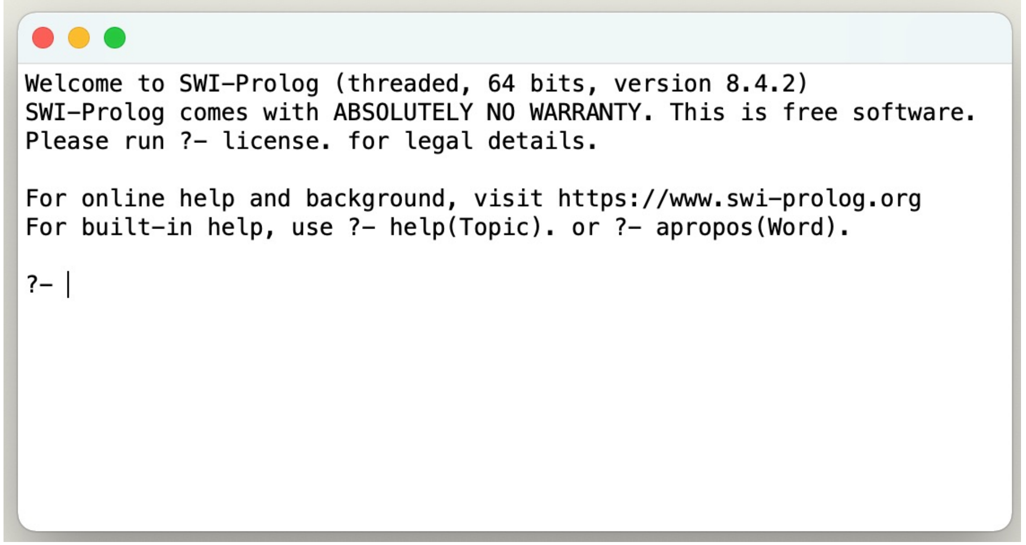
https://www.swi-prolog.org/download/stable/doc/SWI-Prolog-8.4.2.pdf

SWI Prolog Editor



```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.2)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- |
```

SWI Prolog Editor



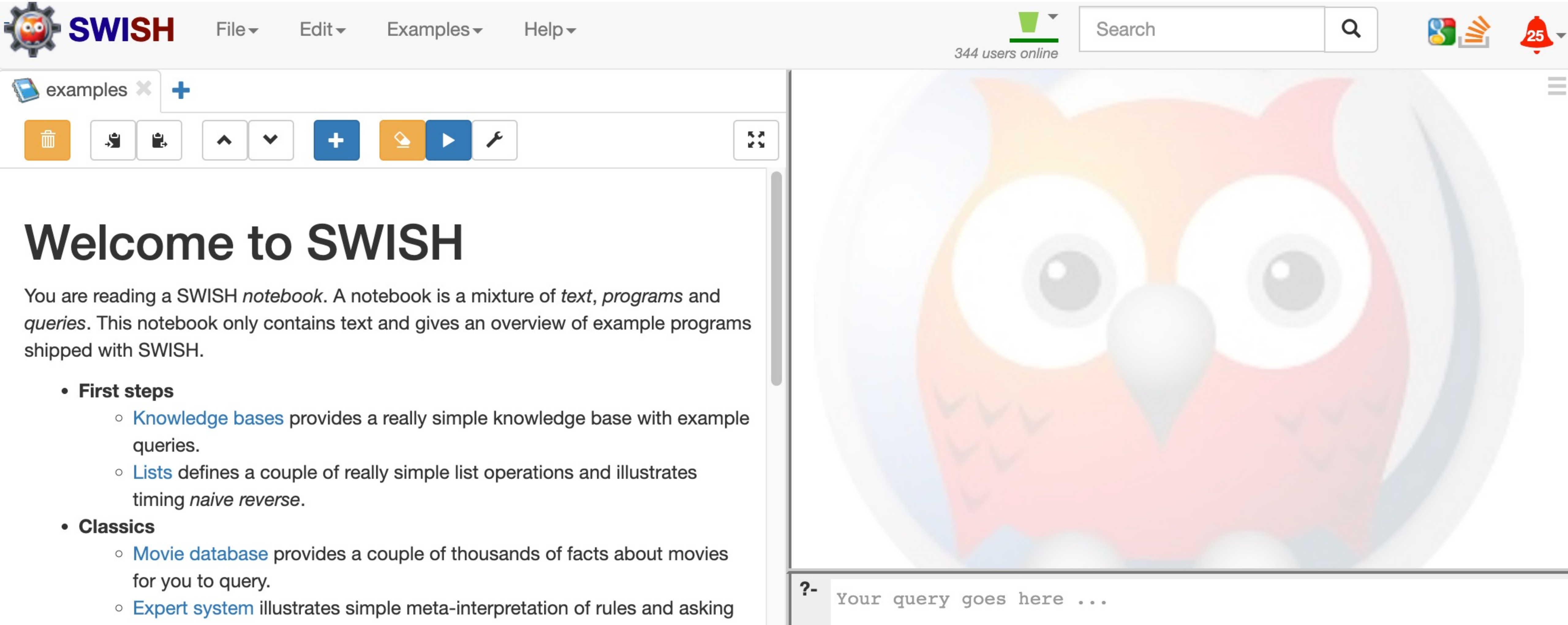
```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.2)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- |
```

working_directory(D,D).

[fileName.pl]

SWI Prolog Editor

https://swish.swi-prolog.org/example/examples.swinb

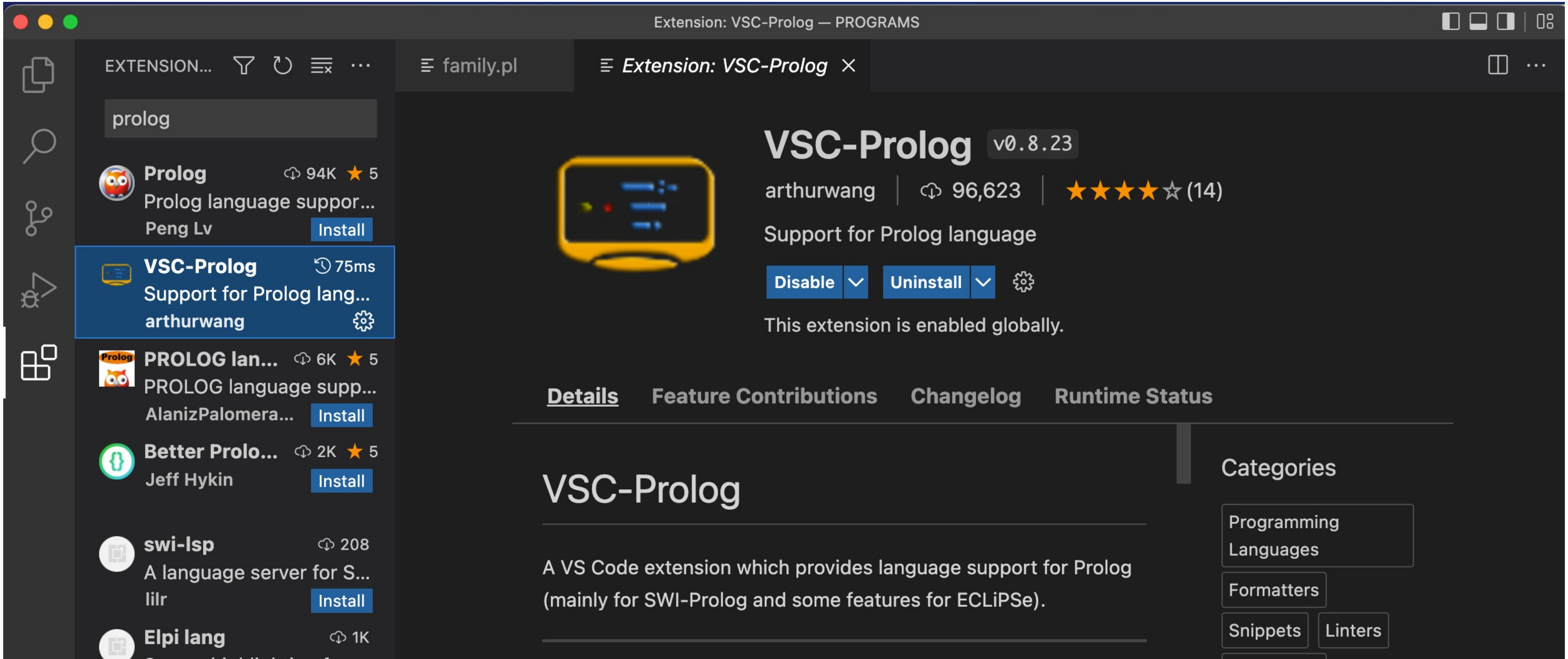


Welcome to SWISH

You are reading a SWISH *notebook*. A notebook is a mixture of *text*, *programs* and *queries*. This notebook only contains text and gives an overview of example programs shipped with SWISH.

- **First steps**
  - Knowledge bases provides a really simple knowledge base with example queries.
  - Lists defines a couple of really simple list operations and illustrates timing *naive reverse*.
- **Classics**
  - Movie database provides a couple of thousands of facts about movies for you to query.
  - Expert system illustrates simple meta-interpretation of rules and asking

VSC-Prolog



Extension: VSC-Prolog — PROGRAMS

EXTENSION...

prolog

Prolog 94K ★ 5
Prolog language suppor...
Peng Lv Install

VSC-Prolog 75ms
Support for Prolog lang...
arthurwang

PROLOG lan... 6K ★ 5
PROLOG language supp...
AlanizPalomera... Install

Better Prolo... 2K ★ 5
Jeff Hykin Install

swi-lsp 208
A language server for S...
lilr Install

Elpi lang 1K

family.pl

Extension: VSC-Prolog

VSC-Prolog v0.8.23

arthurwang | 96,623 | ★★★★☆ (14)

Support for Prolog language

Disable | Uninstall

This extension is enabled globally.

Details | Feature Contributions | Changelog | Runtime Status

VSC-Prolog

A VS Code extension which provides language support for Prolog (mainly for SWI-Prolog and some features for ECLiPSe).

Categories

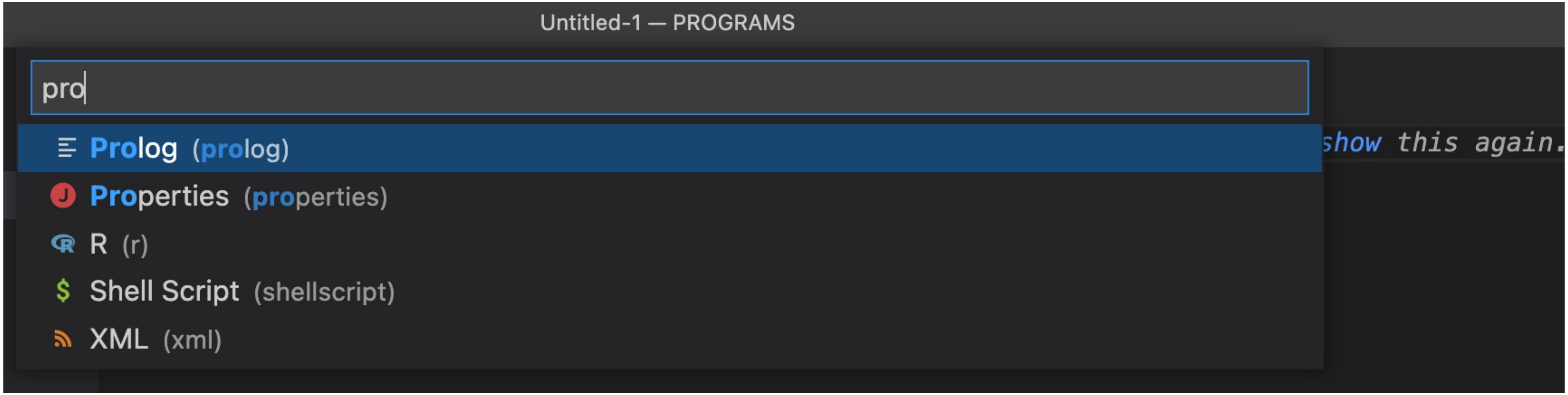
Programming Languages

Formatters

Snippets

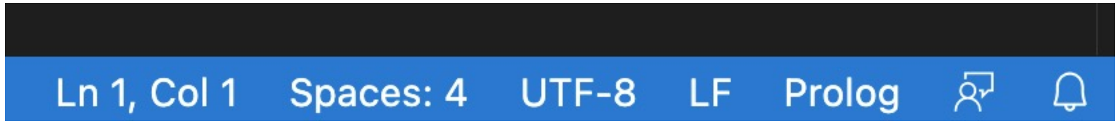
Linters

VSC-Prolog



Untitled-1 — PROGRAMS

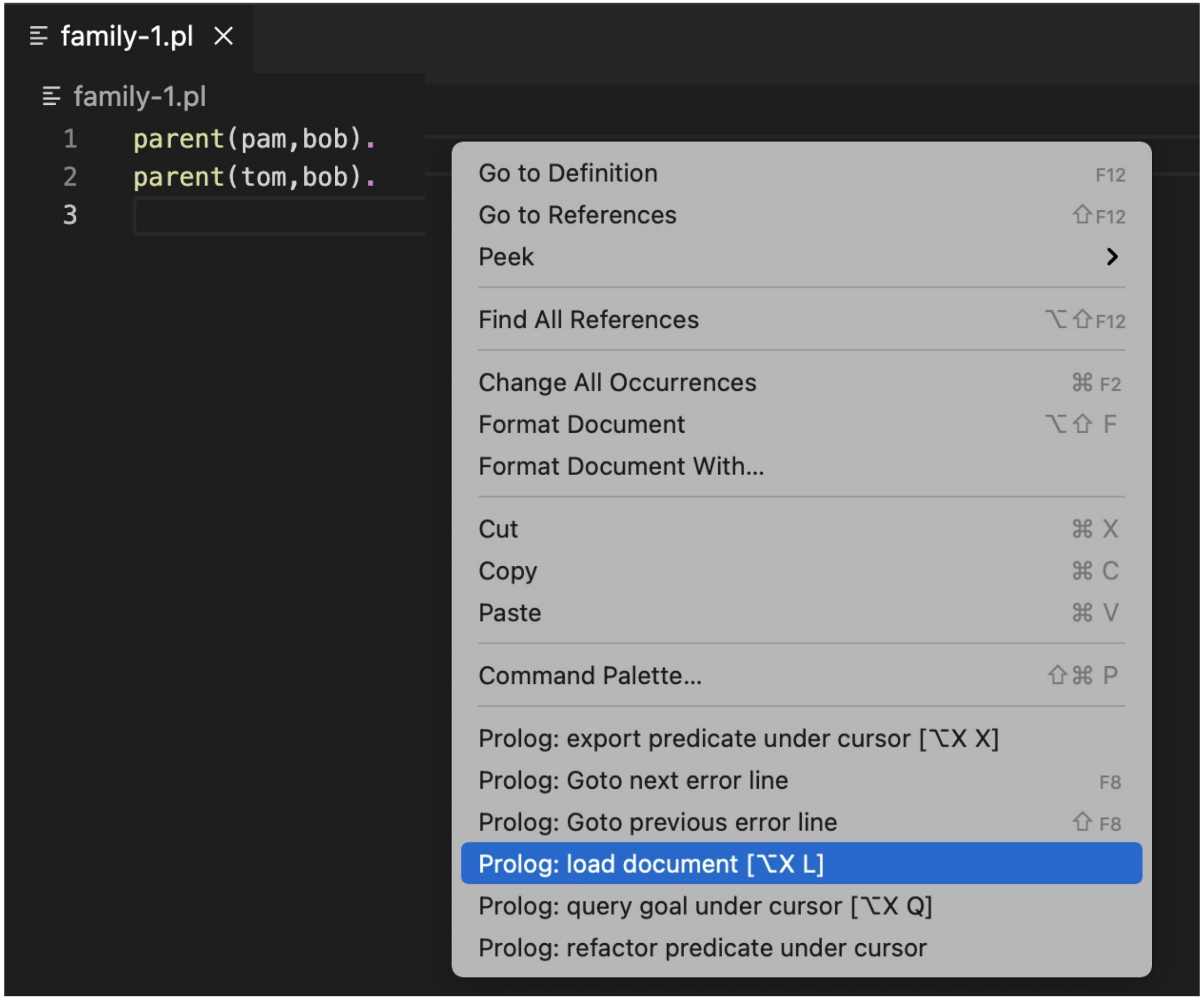
pro



Ln 1, Col 1   Spaces: 4   UTF-8   LF   Prolog

VSC-Prolog

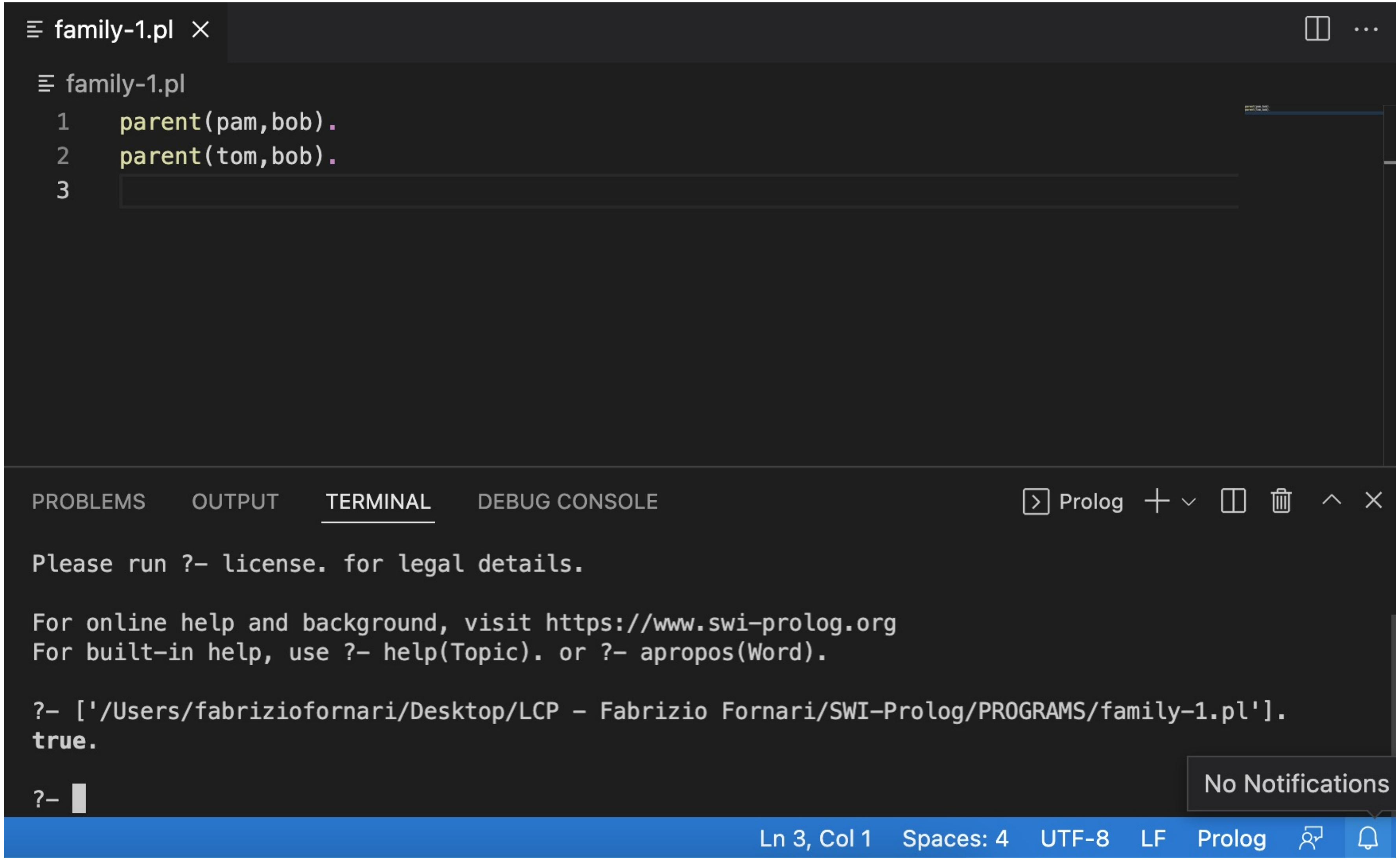
parent(pam,bob).
parent(tom,bob).



```
parent(pam,bob).
parent(tom,bob).

```

Go to Definition F12
Go to References ⇧F12
Peek ›
Find All References ⌥⇧F12
Change All Occurrences ⌘ F2
Format Document ⌥⇧ F
Format Document With...
Cut ⌘ X
Copy ⌘ C
Paste ⌘ V
Command Palette... ⇧⌘ P
Prolog: export predicate under cursor [⌥X X]
Prolog: Goto next error line F8
Prolog: Goto previous error line ⇧ F8
Prolog: load document [⌥X L]
Prolog: query goal under cursor [⌥X Q]
Prolog: refactor predicate under cursor

VSC-Prolog



```prolog
parent(pam,bob).
parent(tom,bob).

```

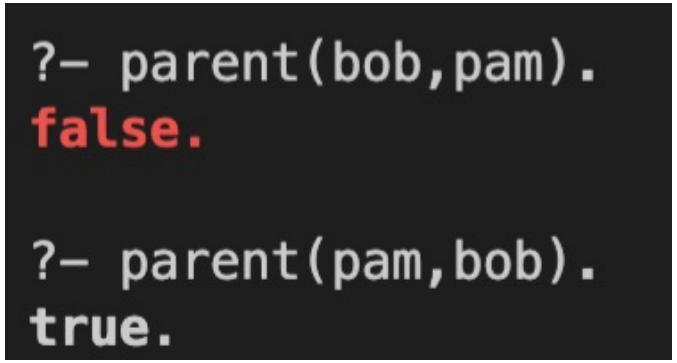
```
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- ['/Users/fabriziofornari/Desktop/LCP - Fabrizio Fornari/SWI-Prolog/PROGRAMS/family-1.pl'].
true.

?- 
```

Prolog

parent(pam,bob).
parent(tom,bob).

Let us ask questions:

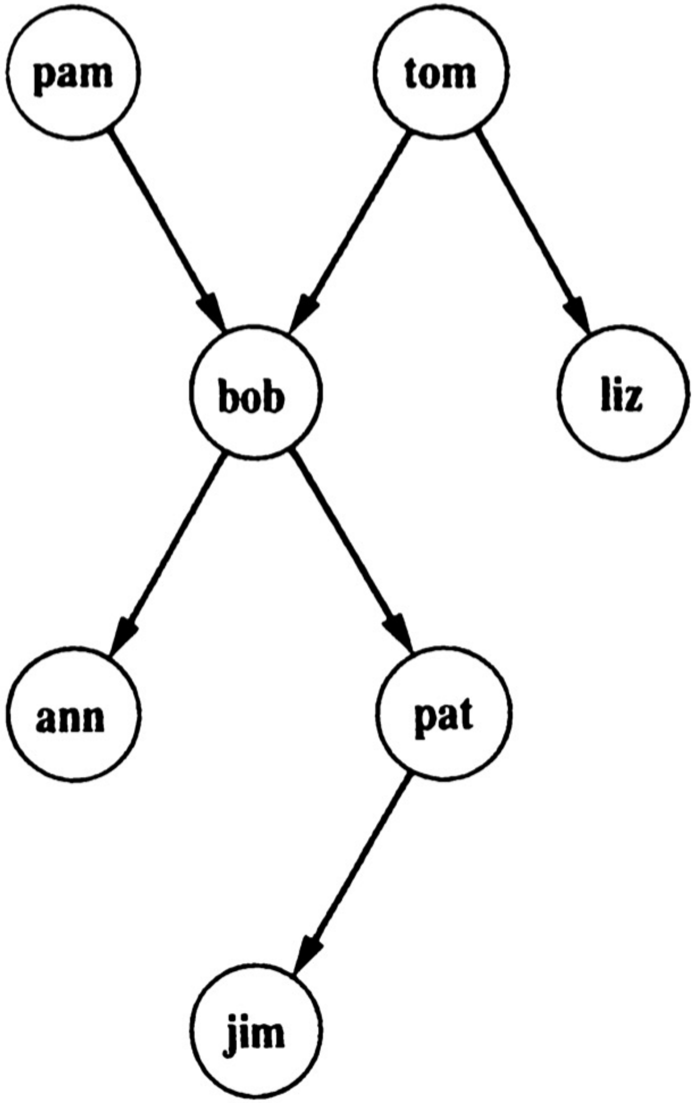
?- parent(bob,pam).
?- parent(pam,bob)



```
?- parent(bob,pam).
false.

?- parent(pam,bob).
true.
```

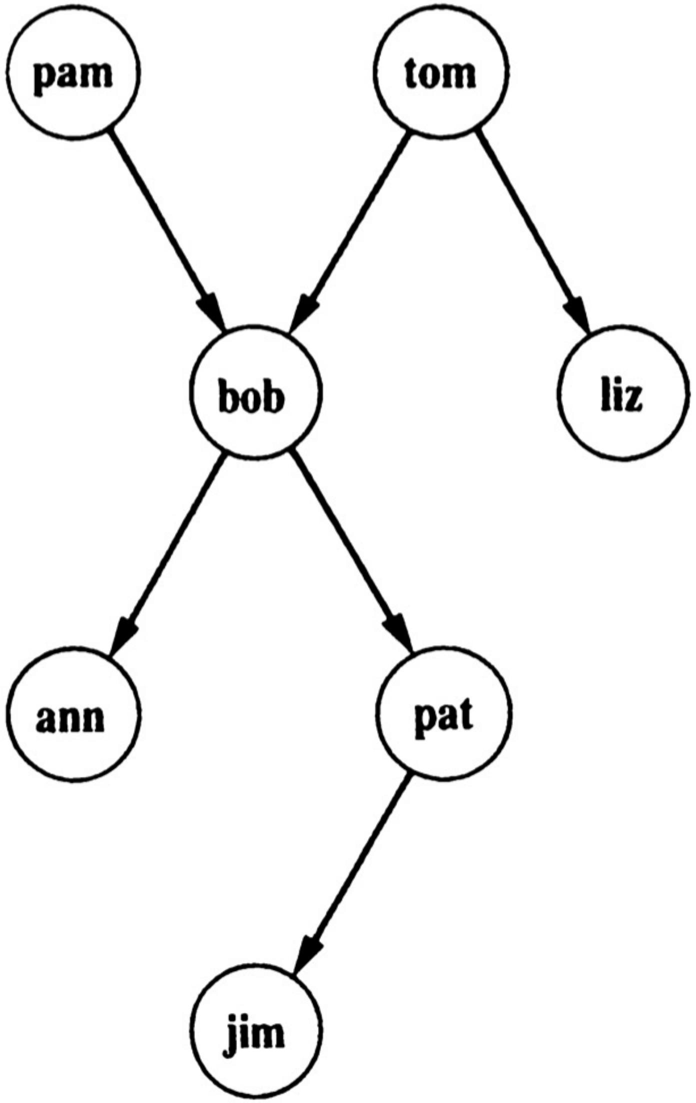
Prolog

```
parent(pam,bob).
parent(tom,bob).
…?
```



Prolog

```
parent(pam,bob).
parent(tom,bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).
```



Prolog

Let us ask questions:

```
?- parent(liz,pat).
?- parent(tom,ben).
```

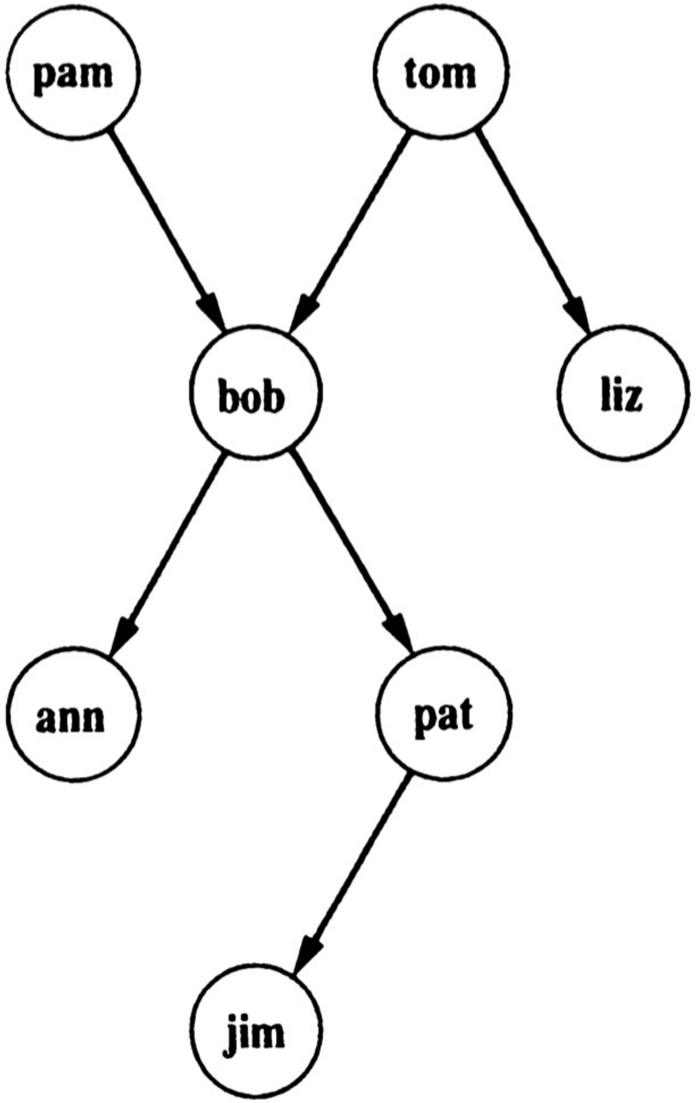
```
?- parent(X,liz).
```

Who is a parent of liz?

```
?- parent(bob,X).
```

How many results?

Let us write a semicolon ;
to display other results



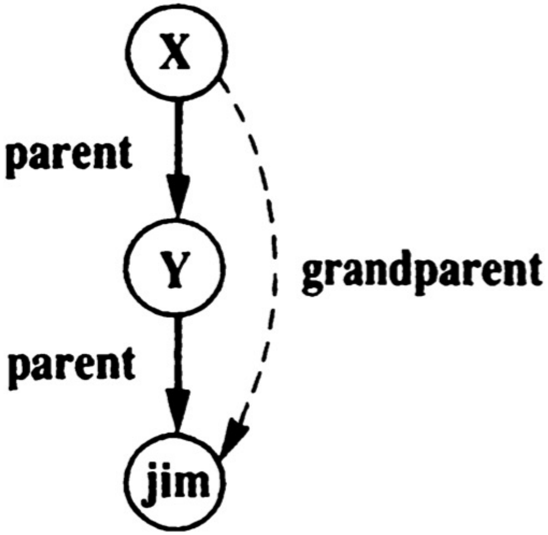
Prolog

Let us ask questions:

Who is a parent of whom?
?- parent(X,Y).

Who is the grandparent of jim?
?- parent(Y,jim),parent(X,Y).

?- parent(X,Y),parent(Y,jim).


X
parent
Y
grandparent
parent
jim