University of BRISTOL

Artificial Intelligence: Logic Programming I

Oliver Ray

bristol.ac.uk

Remember to Check-in

The quick and easy way to register your attendance.

- Download the University of Bristol Check-in app
- Check location is enabled on device and app
- Log in
- Click on class in ‘Check in Now’
- And then click Check-in and Finish

App

About - webpage

Make sure your attendance is registered for all your on-campus classes

History of programming languages (simplified)

Procedural: execute instruction sequences from a given state

Operational style**:** state → state

Logical: search for answers to queries with respect to relational constraints

Axiomatic style: program |= consequences

1950s

Functional: evaluate function definitions on given arguments

Denotational style: input→output

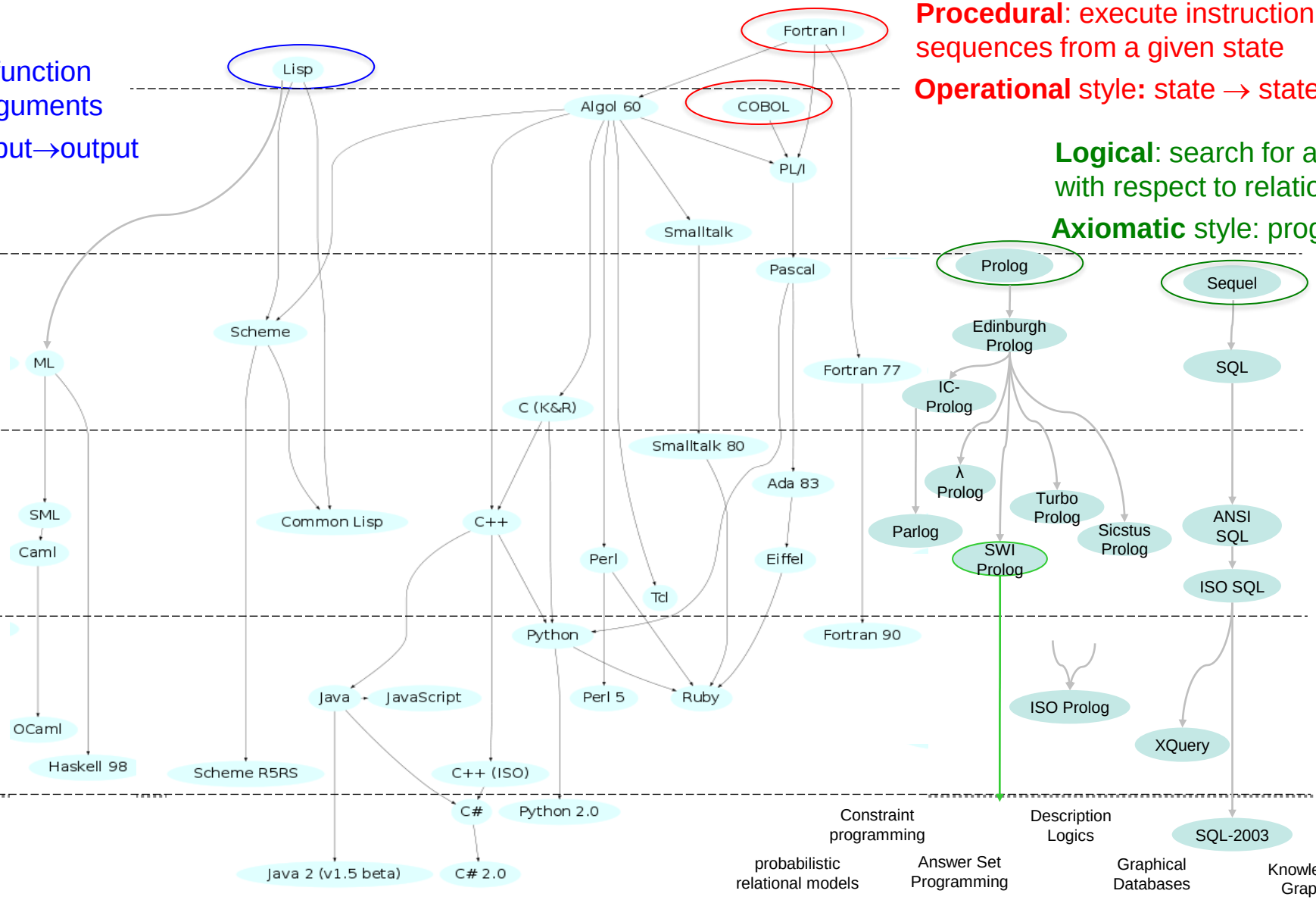
1960s

1970s

1980s

1990s

2000s



PROgrammation en LOGique

Prolog is the most common logic programming language

- Simple syntax, no typing, homoiconic, queries may return any number of answers: none, one or many
- It focuses effort on problem definition, is great for proto-typing and has powerful language processing abilities
- It is also the basis of a many powerful extensions used in real-world tasks: e.g. Constraint Logic Prog. (CLP), Inductive Logic Prog. (ILP), Answer Set Prog. (ASP), …

Datalog

We begin our exploration of Prolog starting from a very simple database perspective (**Datalog**$^{\neg}$) with an example

ACTOR		
Title	Name	Role
american_beauty	kevin_spacey	lester_burnham
...	...	...

ACTRESS		
Title	Name	Role
american_beauty	annette_bening	carolyn_burnham
...	...	...

MOVIE	
Title	Year
american_beauty	1999
anna	1987
...	...

DIRECTOR	
Title	Director
american_beauty	sam_mendes
anna	yurek_bogayevicz
...	...

Qu: Who directed a movie released after 2000 which they also acted in?

Datalog

There are three key parts of a Prolog program:

- Prolog facts → relational database

```
movie(american_beauty,1999).
```

  - one predicate per table; one fact per row
- Prolog rules → relational views

```
released_after(M,Y) :- movie(M,Z), Z>Y.
```

  - intentional definition; materialised when needed
- Prolog queries → relational algebra (RA)

```
?- actor(_, A, _), director(_, A).
```

  - but Prolog is ***much*** easier than RA! (or SQL)

```
?- ( actor(M, A, _) ; actress(M, A, _) ) , director(M, A), released_after(M,1999) .
```

project union join select

SWI-Prolog

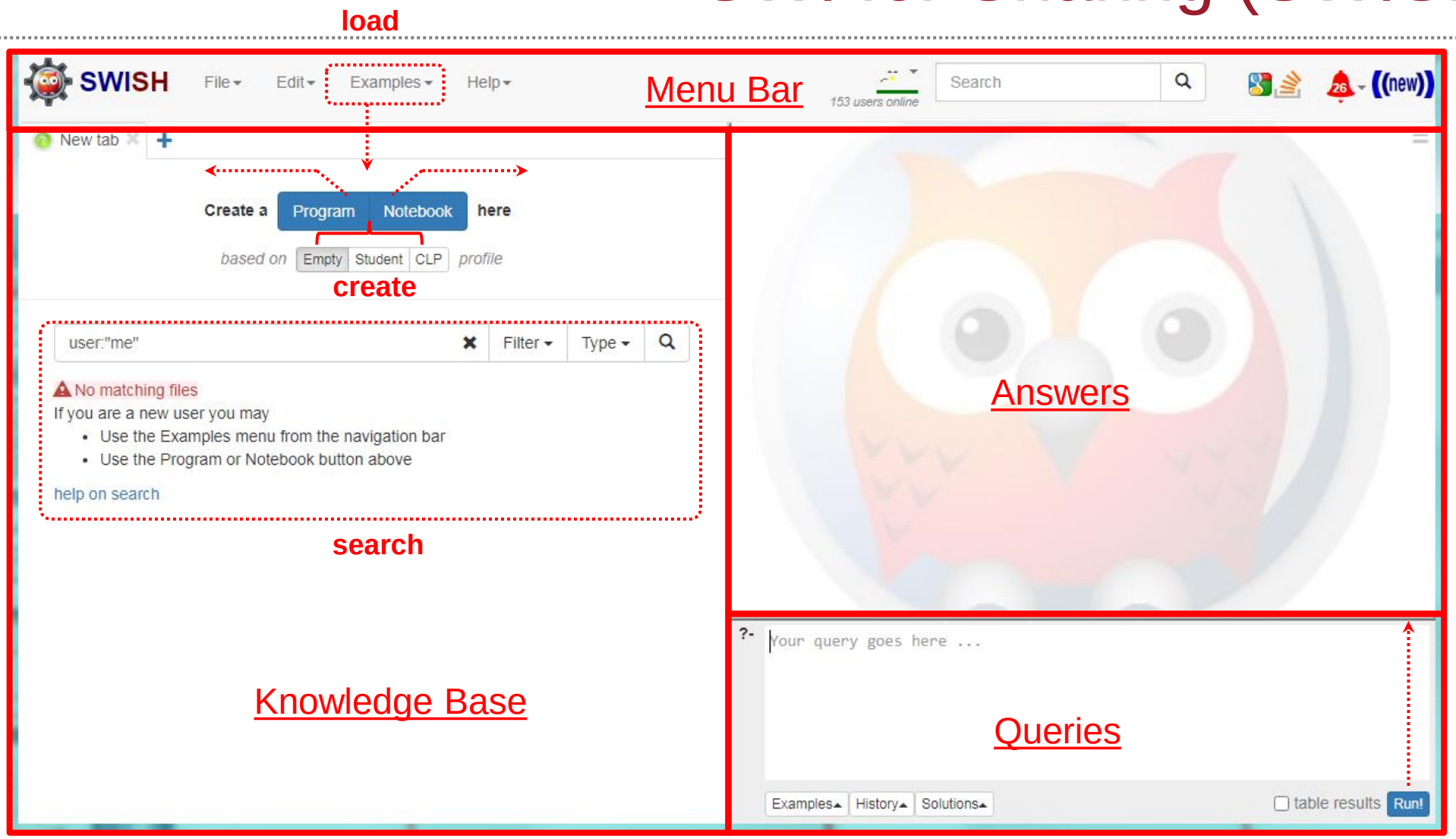
SWI-Prolog is a popular, well-supported, free Prolog system

- SWIPL Engine + SWISH IDE
- Easy to install on Linux, Mac and Windows
- Hosted as a sand-boxed web-service
- Pre-installed on CS lab machines
- User-friendly with lots of examples
- https://www.swi-prolog.org/

All these methods will be suitable for the week 1 lab, but weeks 2&3 will be best done with local installation

https://www.swi-prolog.org/pldoc/man?section=implhistory

SWI for Sharing (SWISH)



Can run locally or on hosted server at https://swish.swi-prolog.org/

Example Program

SWISH File Edit Examples Help

154 users online

Search

examples

Example programs

Prolog tutorials

SWISH tutorials

Usage statistics

Welcome to SW

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 for missing knowledge.
  - Eliza implements the classical shrink.
  - English grammar DCG rules for parsing some simple English sentences and show the result as an SVG tree.
- **Puzzles and constraints**
  - Einstein's Riddle A famous puzzle attributed to Einstein.
  - N-Queens (traditional) solves the N-queens problem using traditional Prolog and illustrates domain-specific (graphics) output.
  - N-Queens (`clp(fd)`) as above, illustrating the value of constraint programming.

?- Your query goes here ...

Examples History Solutions table results Run!

Example Program

SWISH File Edit Examples Help 159 users online Search ((new))

examples kb

```
% Some simple test Prolog programs
% -------------------------------

% Knowledge bases

loves(vincent, mia).
loves(marcellus, mia).
loves(pumpkin, honey_bunny).
loves(honey_bunny, pumpkin).

jealous(X, Y) :-
    loves(X, Z),
    loves(Y, Z).


/** <examples>

?- loves(X, mia).
?- jealous(X, Y).

*/
```

in-line comments:
% …

facts:
predicate(terms)

rules:
head :- body

block comments
/* … */

(example) queries
VaRS, cOnsts

?- Your query goes here ...

loves(X, mia).
jealous(X, Y).

Add current query to examples

Examples History Solutions table results Run!

Example Program

SWISH File Edit Examples Help 167 users online Search

examples kb

```
% Some simple test Prolog programs
% -------------------------------

% Knowledge bases

loves(vincent, mia).
loves(marcellus, mia).
loves(pumpkin, honey_bunny).
loves(honey_bunny, pumpkin).

jealous(X, Y) :-
    loves(X, Z),
    loves(Y, Z).


/** <examples>

?- loves(X, mia).
?- jealous(X, Y).

*/
```

loves(X, mia).

X = vincent

Next 10 100 1,000 Stop

(; or enter)

?- loves(X, mia).

(ctrl-enter)

Examples History Solutions table results Run!

Example Program

SWISH File Edit Examples Help 167 users online Search

examples kb

```
% Some simple test Prolog programs
% -------------------------------

% Knowledge bases

loves(vincent, mia).
loves(marcellus, mia).
loves(pumpkin, honey_bunny).
loves(honey_bunny, pumpkin).

jealous(X, Y) :-
    loves(X, Z),
    loves(Y, Z).


/** <examples>

?- loves(X, mia).
?- jealous(X, Y).

*/
```

loves(X, mia).

X = vincent
X = marcellus

?- loves(X, mia).

Examples History Solutions table results Run!

Example Program

SWISH File Edit Examples Help 162 users online Search

examples kb

```
% Some simple test Prolog programs
% -------------------------------

% Knowledge bases

loves(vincent, mia).
loves(marcellus, mia).
loves(pumpkin, honey_bunny).
loves(honey_bunny, pumpkin).

jealous(X, Y) :-
    loves(X, Z),
    loves(Y, Z).


/** <examples>

?- loves(X, mia).
?- jealous(X, Y).

*/
```

loves(X, mia).

X = vincent

X = marcellus

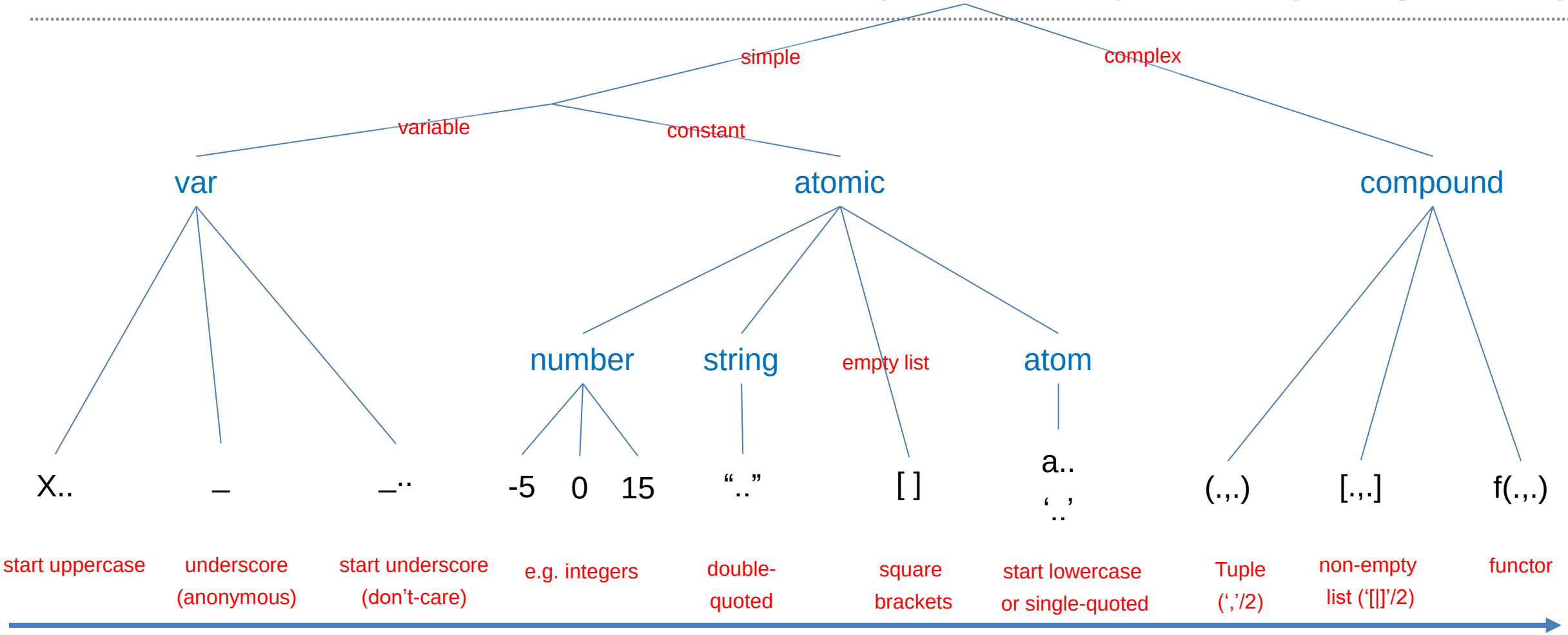
jealous(X, Y).

?

?- jealous(X, Y).

Examples History Solutions table results Run!

Overview of Prolog Term Syntax (simplified)



https://www.swi-prolog.org/pldoc/man?section=standardorder

comparison operators for (ground) numbers: < , > , =< , >= , == , \==

comparison operators for (arbitrary) terms: @< , @> , @=< , @>= , == , \== (take care if comparing variables!)

Actions!

- Have a look around the SWI Prolog website and try to download, install, and run the SWIPL engine and SWISH IDE locally on your computer (highly recommended!)
- This should be relatively easy, but in case of issues, for now try working through this web-hosted SWISH server or try and remotely run SWIPL on the CS lab machines.
- Work through chapters 1 and 3 of the excellent free on-line tutorial Learn Prolog Now! which will take you through the basics of Prolog very simply and effectively

University of BRISTOL

LearnPrologNow! Schedule

Chapter 1	Facts, Rules, and Queries	wk1
Chapter 2	Unification and Proof Search	wk3
Chapter 3	Recursion	wk2
Chapter 4	Lists	wk2
Chapter 5	Arithmetic	wk2
Chapter 6	More Lists	wk3
Chapter 7	Definite Clause Grammars	not examinable!
Chapter 8	More Definite Clause Grammars	not examinable!
Chapter 9	A Closer Look at Terms	wk1+
Chapter 10	Cuts and Negation	wk2
Chapter 11	Database Manipulation and Collecting Solutions	wk2
Chapter 12	Working With Files	not examinable!

“AI Classics” Opportunities

Optional self-study topics (courtesy of Seth Bullock):

- Week 1: Alan Turing and The Turing Test (‘50s)
- Week 2: Newell & Simon's Physical Symbol Systems Hypothesis (‘70s)
- Week 3: The Chinese Room (‘80s)
- Week 4: Brooks and “Nouvelle AI” (‘90s)

University of BRISTOL

Thank you