NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

Introduction to Prolog – Installation and First Exercises

LABWORK 2

2023 - 2024

MDE – Lab2 Objectives and Scheduling

- ❑ Lab 2 class 1: Introduction, Installation and Exercises
- ❑ Lab 2 class 2: Lab Work Modeling
- ❑ Lab 2 class 3: Lab Work Modeling and Implementation
- ❑ Lab 2 class 4: Lab Work Implementation

❑ **Delivery date: 2024/05/27**

MDE – Lab2 (Class 1)

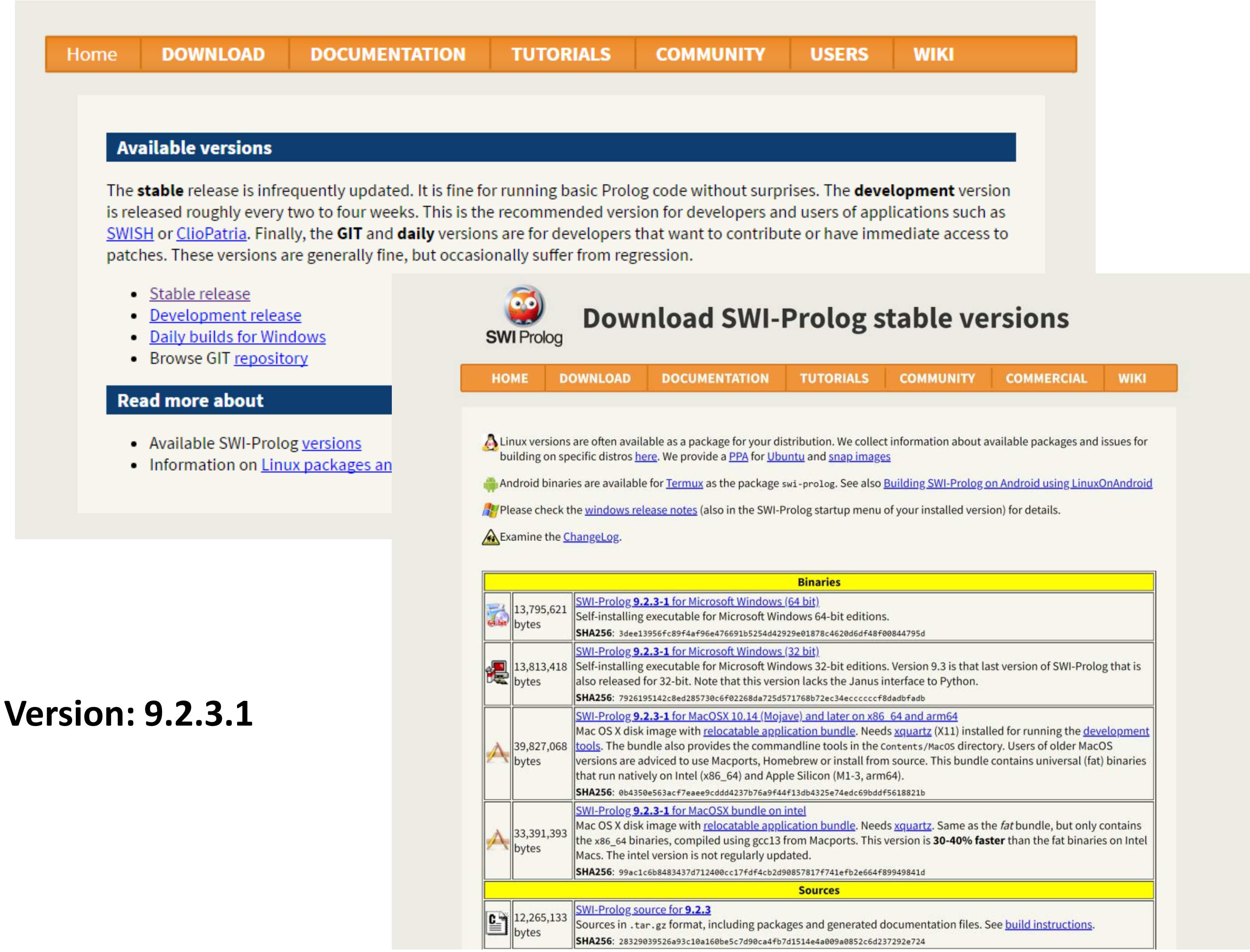
- ❑ Swi-prolog installation

- ❑ Examples with:
  - ❑ Representation of facts and rules
  - ❑ Queries
  - ❑ Recursion

MDE – Lab2: Swi-prolog installation

SWI Prolog

http://www.swi-prolog.org



Available versions

The **stable** release is infrequently updated. It is fine for running basic Prolog code without surprises. The **development** version is released roughly every two to four weeks. This is the recommended version for developers and users of applications such as SWISH or ClioPatria. Finally, the **GIT** and **daily** versions are for developers that want to contribute or have immediate access to patches. These versions are generally fine, but occasionally suffer from regression.

- Stable release
- Development release
- Daily builds for Windows
- Browse GIT repository

Read more about

- Available SWI-Prolog versions
- Information on Linux packages an

Download SWI-Prolog stable versions

Linux versions are often available as a package for your distribution. We collect information about available packages and issues for building on specific distros here. We provide a PPA for Ubuntu and snap images

Android binaries are available for Termux as the package `swi-prolog`. See also Building SWI-Prolog on Android using LinuxOnAndroid

Please check the windows release notes (also in the SWI-Prolog startup menu of your installed version) for details.

Examine the ChangeLog.

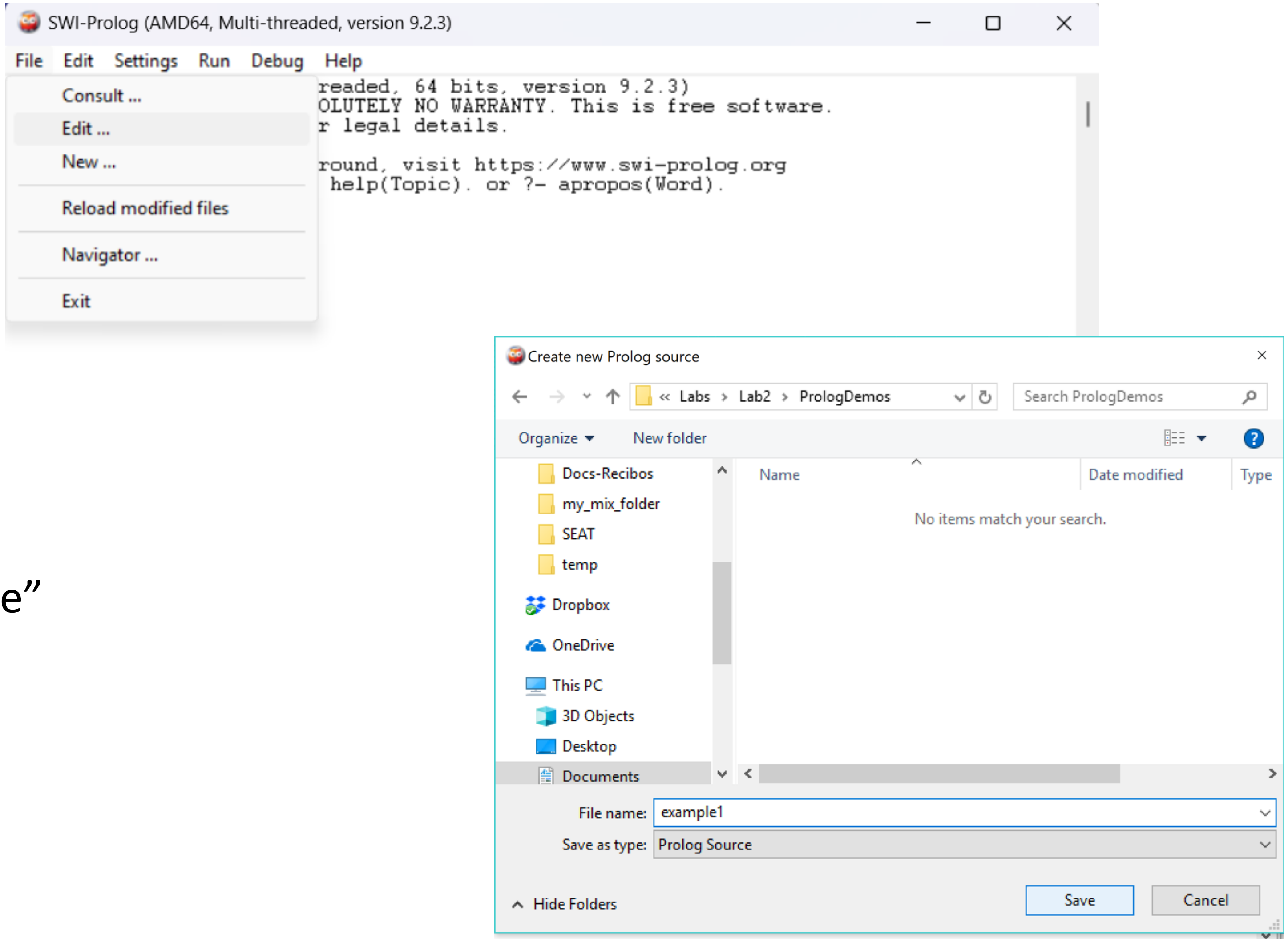
		Binaries
	13,795,621 bytes	SWI-Prolog **9.2.3-1** for Microsoft Windows (64 bit) Self-installing executable for Microsoft Windows 64-bit editions. **SHA256**: `3dee13956fc89f4af96e476691b5254d42929e01878c4620d6df48f00844795d`
	13,813,418 bytes	SWI-Prolog **9.2.3-1** for Microsoft Windows (32 bit) Self-installing executable for Microsoft Windows 32-bit editions. Version 9.3 is that last version of SWI-Prolog that is also released for 32-bit. Note that this version lacks the Janus interface to Python. **SHA256**: `7926195142c8ed285730c6f02268da725d571768b72ec34eccccccf8dadbfadb`
	39,827,068 bytes	SWI-Prolog **9.2.3-1** for MacOSX 10.14 (Mojave) and later on x86_64 and arm64 Mac OS X disk image with relocatable application bundle. Needs xquartz (X11) installed for running the development tools. The bundle also provides the commandline tools in the `Contents/MacOS` directory. Users of older MacOS versions are adviced to use Macports, Homebrew or install from source. This bundle contains universal (fat) binaries that run natively on Intel (x86_64) and Apple Silicon (M1-3, arm64). **SHA256**: `0b4350e563acf7eaee9cddd4237b76a9f44f13db4325e74edc69bddf5618821b`
	33,391,393 bytes	SWI-Prolog **9.2.3-1** for MacOSX bundle on intel Mac OS X disk image with relocatable application bundle. Needs xquartz. Same as the *fat* bundle, but only contains the `x86_64` binaries, compiled using gcc13 from Macports. This version is **30-40% faster** than the fat binaries on Intel Macs. The intel version is not regularly updated. **SHA256**: `99ac1c6b8483437d712400cc17fdf4cb2d90857817f741efb2e664f89949841d`
		Sources
	12,265,133 bytes	SWI-Prolog source for **9.2.3** Sources in `.tar.gz` format, including packages and generated documentation files. See build instructions. **SHA256**: `28329039526a93c10a160be5c7d90ca4fb7d1514e4a009a0852c6d237292e724`

New Stable Version: 9.2.3.1

MDE – Lab2: Using Swi-prolog

http://www.swi-prolog.org

- □ Execute “swipl-win.exe”
- □ Create new .pl file
  - □ File->new



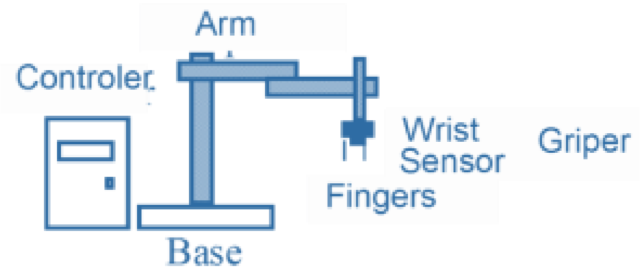
```
readed, 64 bits, version 9.2.3)
OLUTELY NO WARRANTY. This is free software.
r legal details.

round, visit https://www.swi-prolog.org
 help(Topic). or ?- apropos(Word).
```

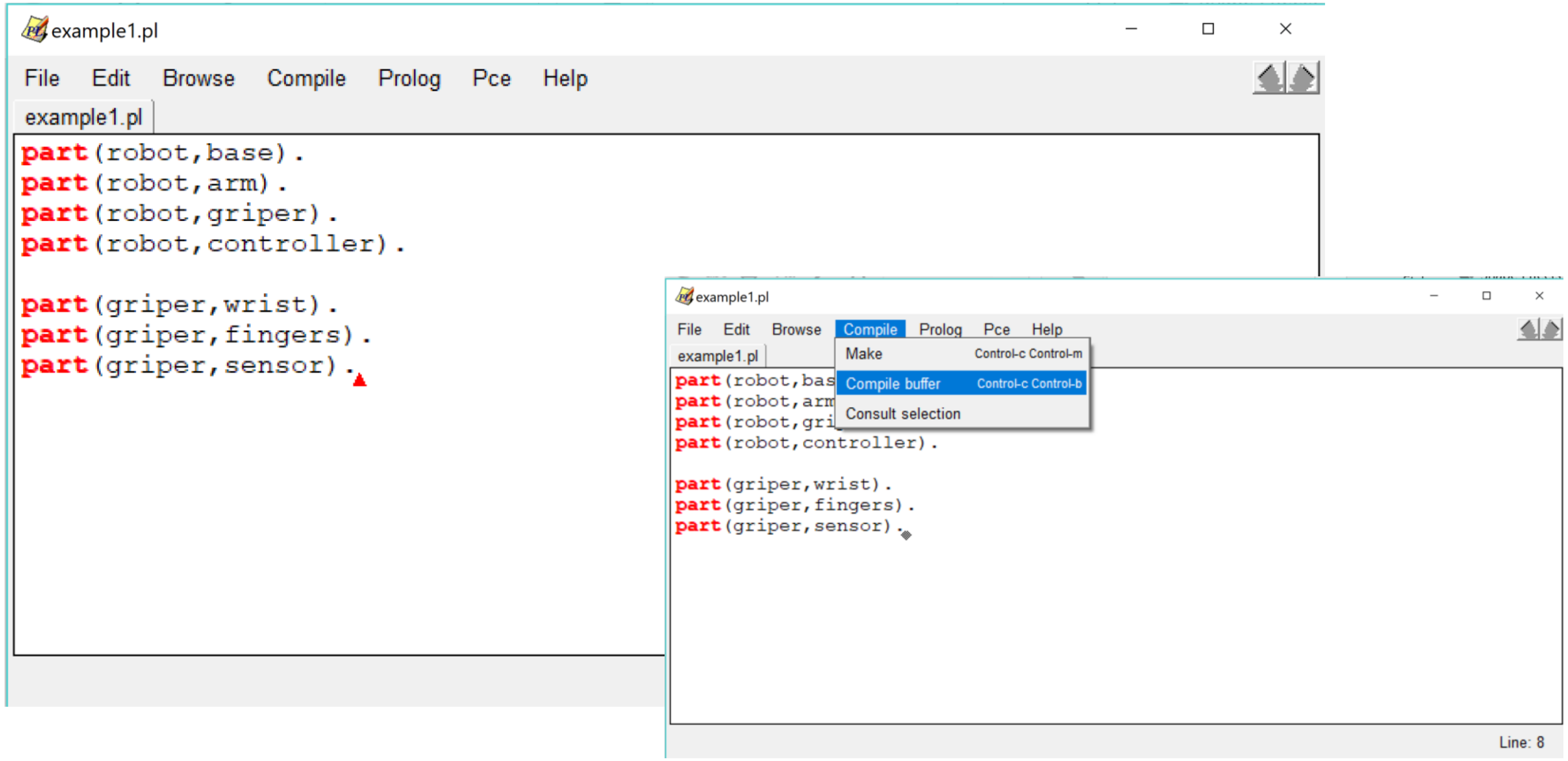
File name: example1

Save as type: Prolog Source

MDE – Lab2: Representation of facts

- Example 1: Model the structure of a robot [Done in the THORETICAL class]




example1.pl
File Edit Browse Compile Prolog Pce Help
example1.pl

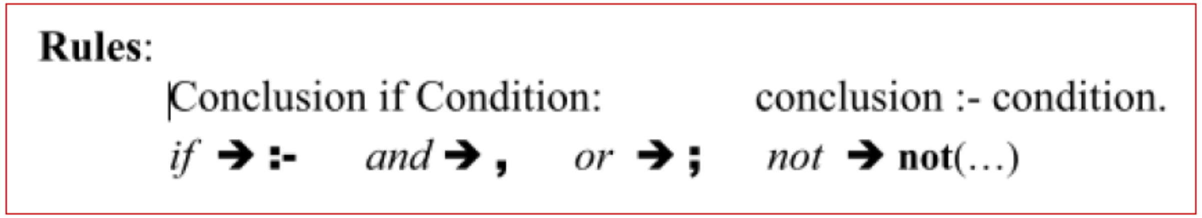
```
part(robot,base).
part(robot,arm).
part(robot,griper).
part(robot,controller).

part(griper,wrist).
part(griper,fingers).
part(griper,sensor).
```

example1.pl
File Edit Browse Compile Prolog Pce Help
example1.pl
Make Control-c Control-m
Compile buffer Control-c Control-b
Consult selection
Line: 8

```
part(robot,bas
part(robot,arm
part(robot,gri
part(robot,controller).

part(griper,wrist).
part(griper,fingers).
part(griper,sensor).
```

MDE – Lab2: Representation of rules

FROM THEORETICAL CLASSES…



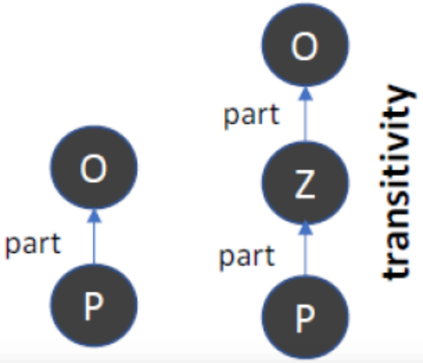
Rules:

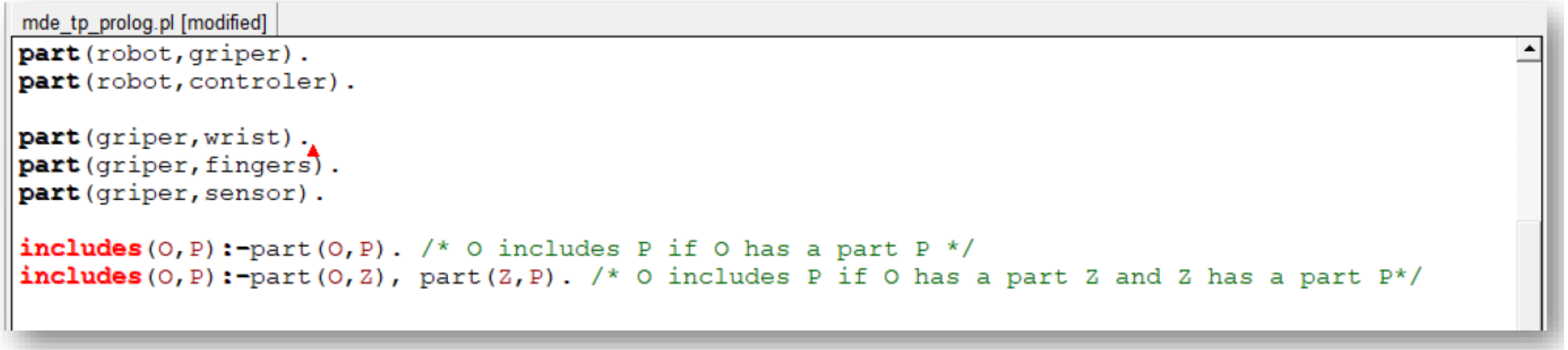
Conclusion if Condition: conclusion :- condition.

if → **:-** *and* → **,** *or* → **;** *not* → **not**(…)

includes(O,P) :- part(O,P). /* O includes P if O has a part P */

includes(O, P) :- part(O,Z), part(Z,P). /* O includes P if O has a part Z and Z has a part P*/

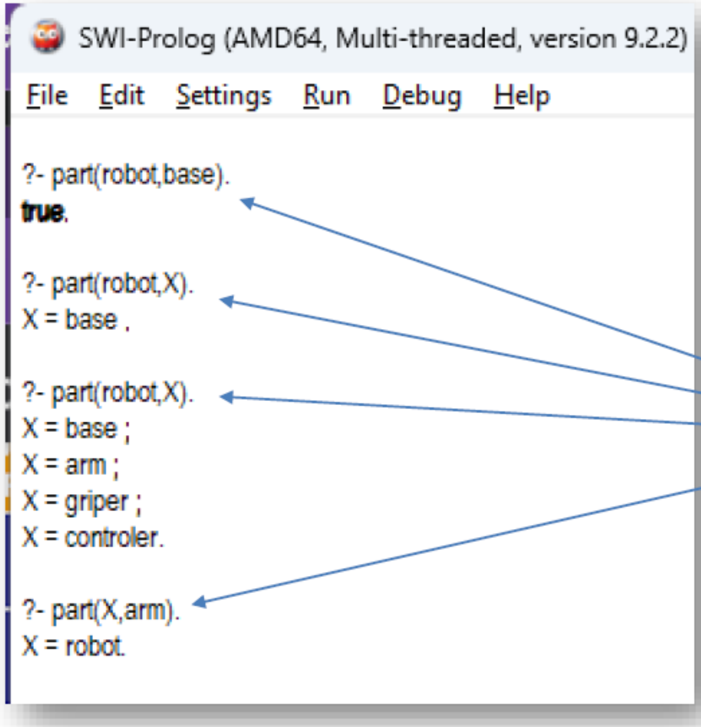




```
part(robot,griper).
part(robot,controler).

part(griper,wrist).
part(griper,fingers).
part(griper,sensor).

includes(O,P):-part(O,P). /* O includes P if O has a part P */
includes(O,P):-part(O,Z), part(Z,P). /* O includes P if O has a part Z and Z has a part P*/
```

MDE – Lab2: Queries

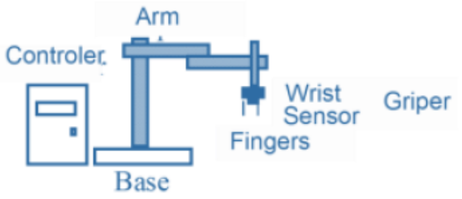
FROM THEORETICAL CLASSES…

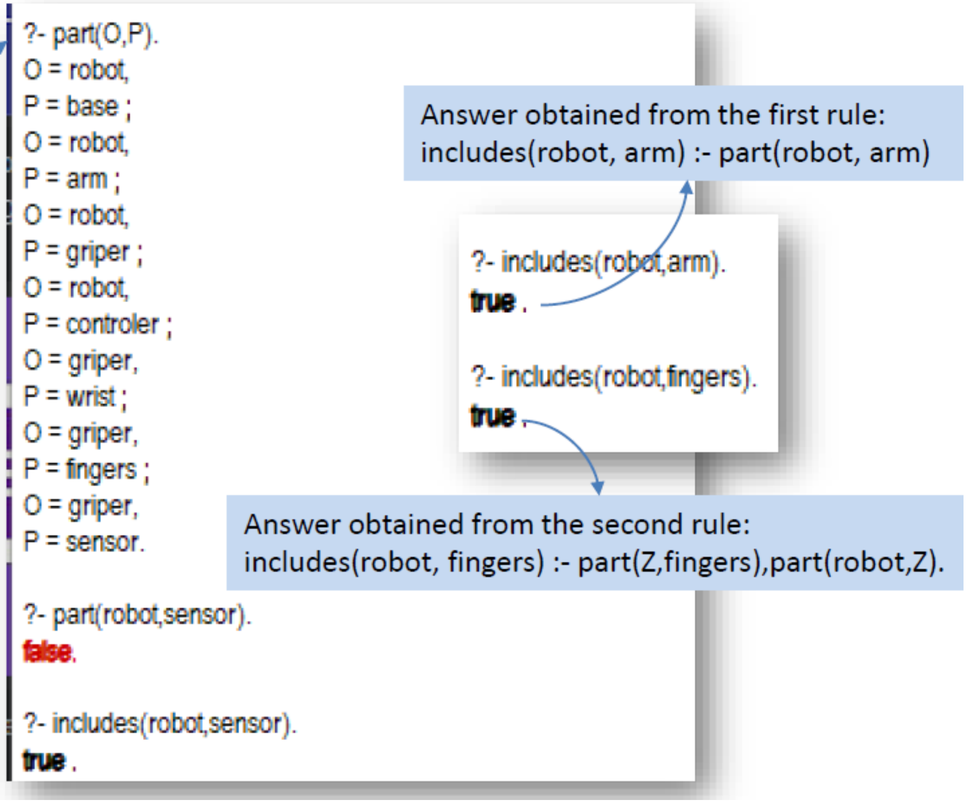


SWI-Prolog (AMD64, Multi-threaded, version 9.2.2)
File Edit Settings Run Debug Help

```
?- part(robot,base).
true.

?- part(robot,X).
X = base .

?- part(robot,X).
X = base ;
X = arm ;
X = griper ;
X = controler.

?- part(X,arm).
X = robot.
```

Examples “?-”





```
?- part(O,P).
O = robot,
P = base ;
O = robot,
P = arm ;
O = robot,
P = griper ;
O = robot,
P = controler ;
O = griper,
P = wrist ;
O = griper,
P = fingers ;
O = griper,
P = sensor.

?- part(robot,sensor).
false.

?- includes(robot,sensor).
true .
```

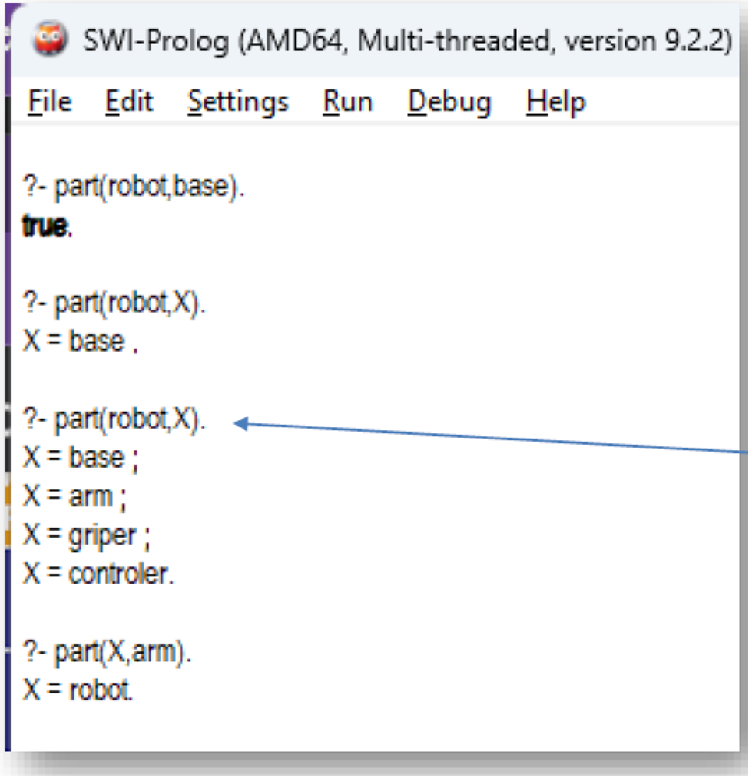
```
?- includes(robot,arm).
true .

?- includes(robot,fingers).
true .
```

Answer obtained from the first rule:
includes(robot, arm) :- part(robot, arm)

Answer obtained from the second rule:
includes(robot, fingers) :- part(Z,fingers),part(robot,Z).

MDE – Lab2: Queries

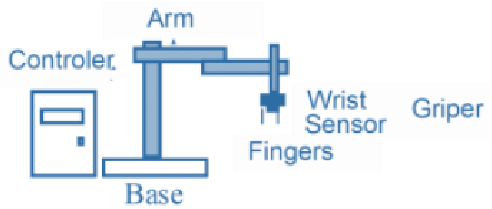
FROM THEORETICAL CLASSES…

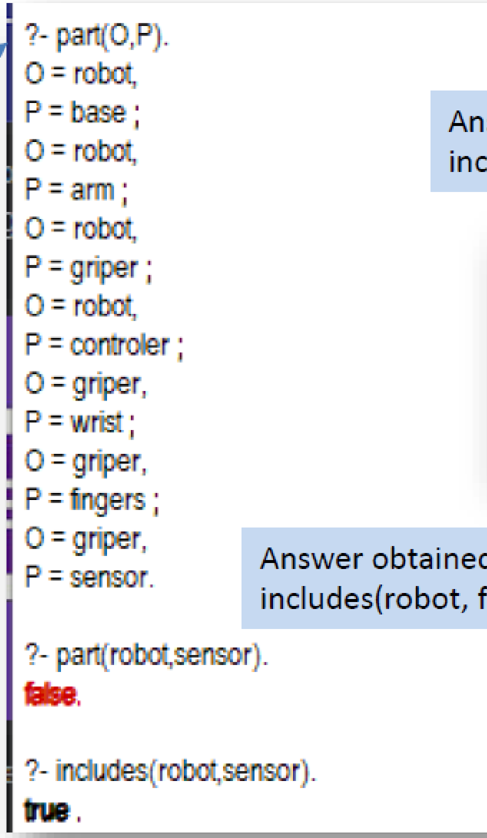


SWI-Prolog (AMD64, Multi-threaded, version 9.2.2)

```
?- part(robot,base).
true.

?- part(robot,X).
X = base .

?- part(robot,X).
X = base ;
X = arm ;
X = griper ;
X = controler.

?- part(X,arm).
X = robot.
```

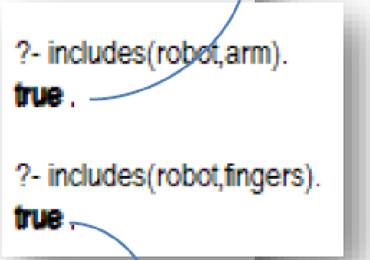
Multiple results





```
?- part(O,P).
O = robot,
P = base ;
O = robot,
P = arm ;
O = robot,
P = griper ;
O = robot,
P = controler ;
O = griper,
P = wrist ;
O = griper,
P = fingers ;
O = griper,
P = sensor.

?- part(robot,sensor).
false.

?- includes(robot,sensor).
true .
```

Answer obtained from the first rule:
includes(robot, arm) :- part(robot, arm)

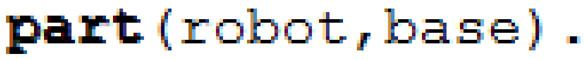


```
?- includes(robot,arm).
true .

?- includes(robot,fingers).
true .
```

Answer obtained from the second rule:
includes(robot, fingers) :- part(Z,fingers),part(robot,Z).

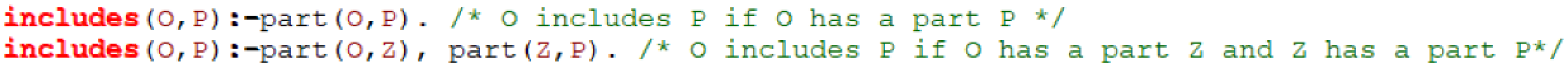
MDE – Lab2: Some notes

- ❑ Facts



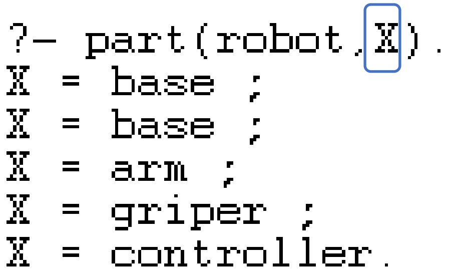
```
part(robot,base).
```

- ❑ Rules



```
includes(O,P):-part(O,P). /* O includes P if O has a part P */
includes(O,P):-part(O,Z), part(Z,P). /* O includes P if O has a part Z and Z has a part P*/
```

- ❑ Variables



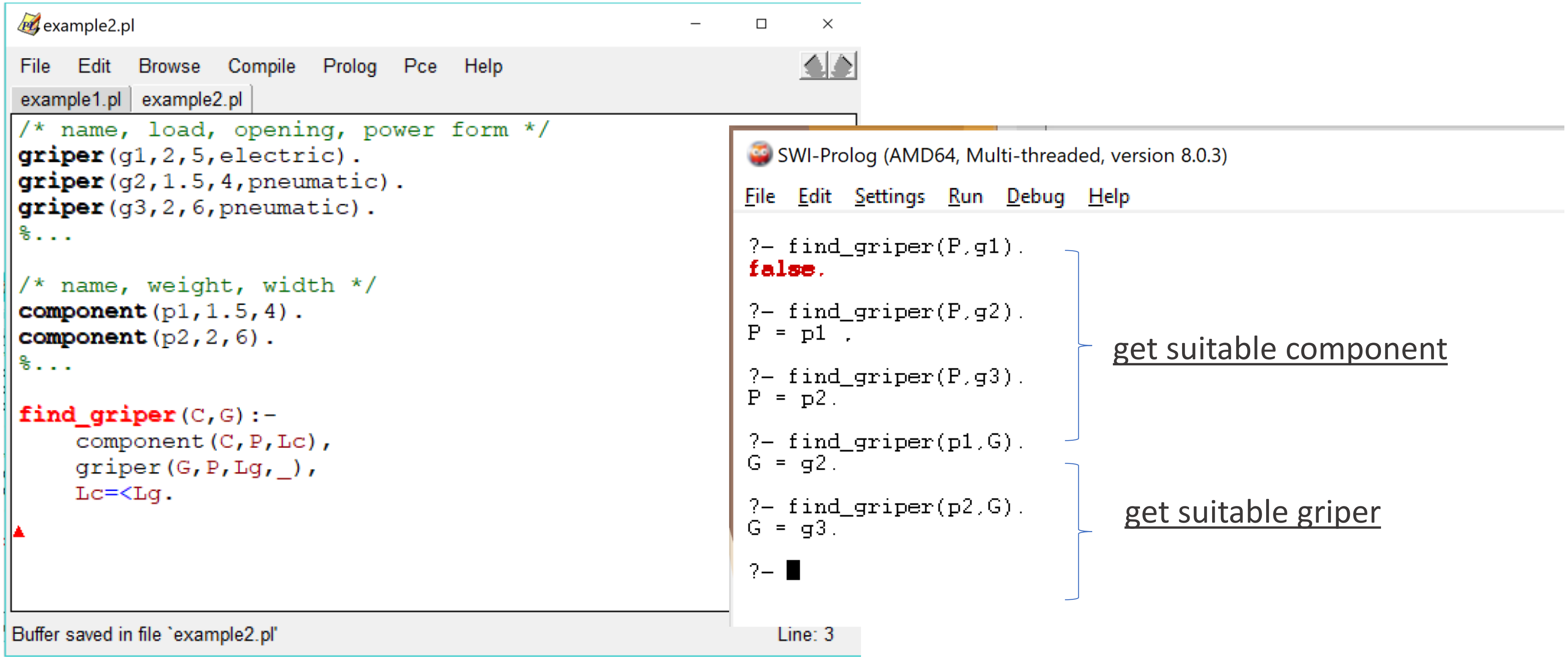
```
?- part(robot,X).
X = base ;
X = base ;
X = arm ;
X = griper ;
X = controller.
```

Capital letter

Lower case -> Constants

MDE – Lab2: another example (i)

- Example 2: Robot components [Done in the THORETICAL class]



```
/* name, load, opening, power form */
griper(g1,2,5,electric).
griper(g2,1.5,4,pneumatic).
griper(g3,2,6,pneumatic).
%...

/* name, weight, width */
component(p1,1.5,4).
component(p2,2,6).
%...

find_griper(C,G):-
    component(C,P,Lc),
    griper(G,P,Lg,_),
    Lc=<Lg.
```

Buffer saved in file `example2.pl'

Line: 3

```
?- find_griper(P,g1).
false.

?- find_griper(P,g2).
P = p1 .

?- find_griper(P,g3).
P = p2.

?- find_griper(p1,G).
G = g2.

?- find_griper(p2,G).
G = g3.

?- 
```

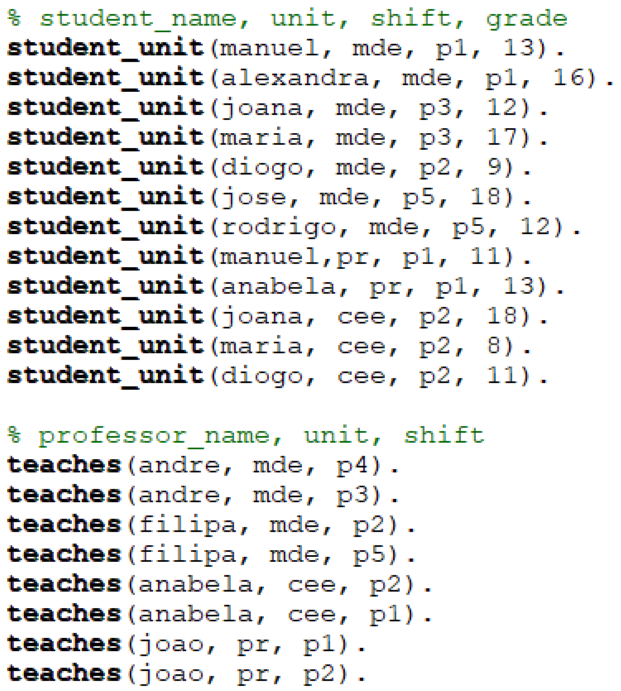
get suitable component

get suitable griper

MDE – Lab2

❑ Exercise 1:

Consider the following predicates:



```
% student_name, unit, shift, grade
student_unit(manuel, mde, p1, 13).
student_unit(alexandra, mde, p1, 16).
student_unit(joana, mde, p3, 12).
student_unit(maria, mde, p3, 17).
student_unit(diogo, mde, p2, 9).
student_unit(jose, mde, p5, 18).
student_unit(rodrigo, mde, p5, 12).
student_unit(manuel,pr, p1, 11).
student_unit(anabela, pr, p1, 13).
student_unit(joana, cee, p2, 18).
student_unit(maria, cee, p2, 8).
student_unit(diogo, cee, p2, 11).

% professor_name, unit, shift
teaches(andre, mde, p4).
teaches(andre, mde, p3).
teaches(filipa, mde, p2).
teaches(filipa, mde, p5).
teaches(anabela, cee, p2).
teaches(anabela, cee, p1).
teaches(joao, pr, p1).
teaches(joao, pr, p2).
```

Write the corresponding rules that would answer the following queries:

a) Which students are enrolled in shift p3?

b) Which students from p5 have a grade > 14?

c) Which units is diogo enrolled in?

d) Who are the students of professor andre?

e) What are the professors of student joana?

MDE – Lab2

- Exercise 2:

Given the following facts:

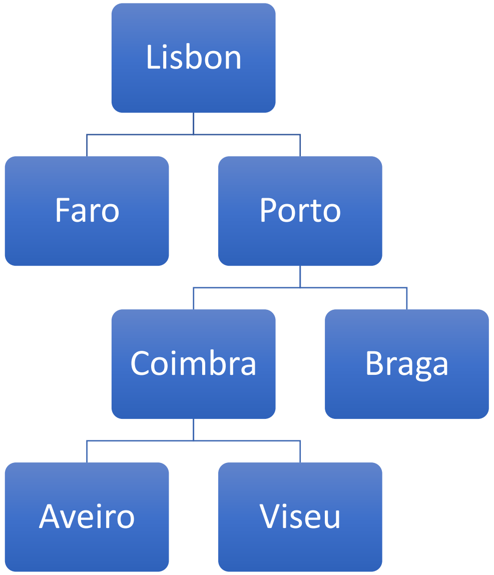
- Maria is fatter than Ana
- Ana is fatter than Luisa
- Luisa is fatter than Diana
- Diana is fatter than Sara

Write the corresponding facts and rules (using recursion) that determine that Maria is heavier that Sara.

MDE – Lab2

Exercise 3:

Consider the following road tree that connects several cities in Portugal (one direction)



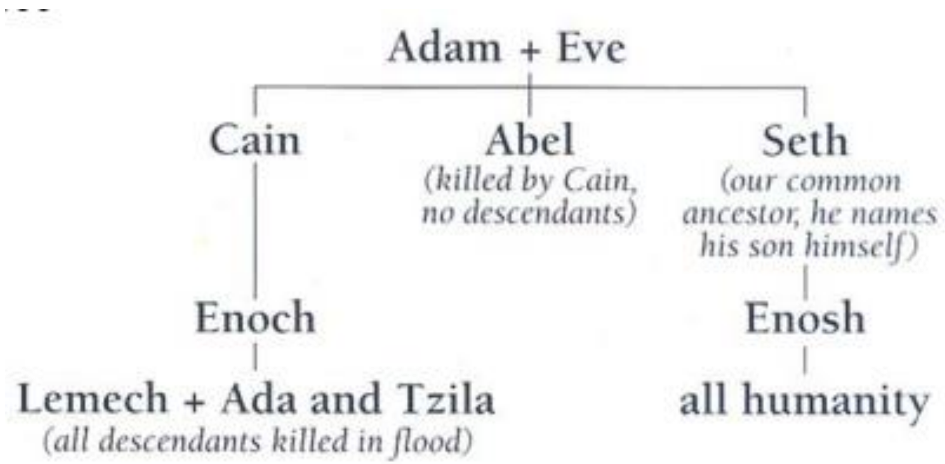
Write the corresponding facts and rules (using recursion) that would answer the following queries:

a) Can I travel from Lisbon to Viseu?
b) Can I travel from Faro to Braga?
c) How many cities I cross between Lisbon and Aveiro?

MDE – Lab2

- Exercise 4:

Use the predicates **father/2** and **mother/2** to represent the genealogic tree (in this example we use the Adam and Eve's genealogic tree, but you can use yours!):



Adam + Eve

Cain

Abel
(killed by Cain, no descendants)

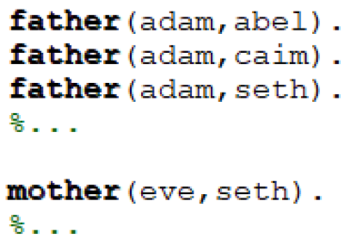
Seth
(our common ancestor, he names his son himself)

Enoch

Enosh

Lemech + Ada and Tzila
(all descendants killed in flood)

all humanity



```
father(adam,abel).
father(adam,caim).
father(adam,seth).
%...

mother(eve,seth).
%...
```

Create rules to capture the following relationships:

- **son**(Father, Mother)
- **grandfather**(Grandfather, Grandson)
- **brother**(Brother1, Brother2)
- **uncle**(Uncle, Nephew)
- **cousin**(Cousin1, Cousin2)
- **ascendant**(Ascendant, Descendant)
- **descendant**(Descendant, Ascendant)

GOOD WORK!