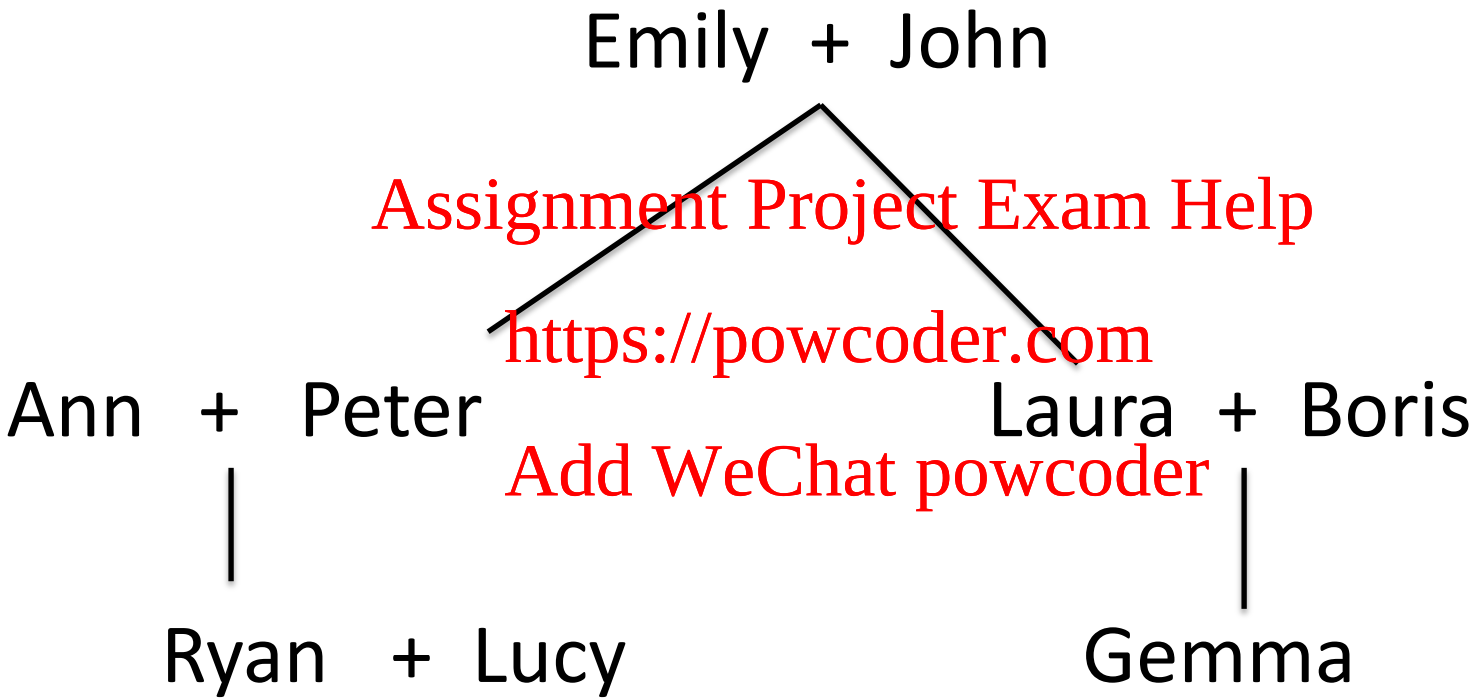
Good vs Bad style of Presenting Prolog Programs

F. Sadri

“Family” Exercise



- Write down facts defining who is
  - (1) female
  - (2) male and
  - (3) who is the child of whom.
- Write a predicate that denotes the *uncle* relation.
- Write a predicate that denotes the *aunt* relation.

BAD Style answers to the Family-Exercise

% by StudentFirstName StudentLastName
% Day Month Year
%%

female(emily).
female(gemma).
child(peter, john).
child(laura, emily) .

female(laura). female(ann).
female(lucy).

```
child(laura,john).
child(ryan, ann).
male(peter).
male(boris ). child( gemma, laura ). male(  ryan).
male( john)   .
        child(peter, emily ).

child(ryan        , peter).
child(gemma, boris      ).
uncle(X,Y):-child(Y,Z),child(Z,A),
child(X,A),X\=Z,male(X).
aunt(X, W) :-child(W,Z),child(Z,A), child(X,A),
X\=Z,female(X).
```

Good Style answers to the Family-Exercise

% % with thanks to Claudia Schulz

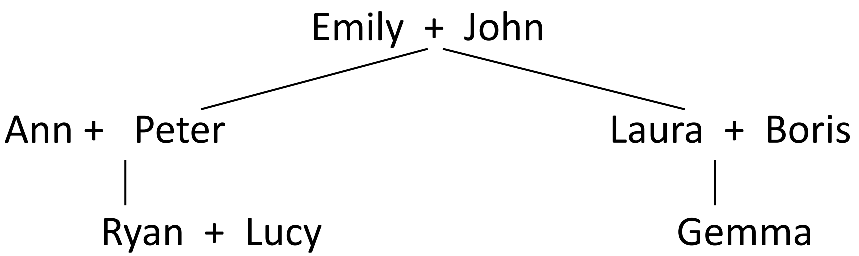
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

STEP 1

- consistent use of whitespaces
- all clauses of one predicate together
- different predicates are separated by spaces
- every clause begins on a new line

STEP 2

- comments to explain the predicates and their arguments
- predicates have sensible names
- document structure




```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% the family database
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% all females - ordered breadth-first
female(emily).
female(ann).
female(laura).
female(lucy).
female(gemma).

% all males - ordered depth-first
male(john).
male(peter).
male(ryan).
male(boris).
```

```
% is_child_of(Child, Parent) means that Child is the child
% of Parent
% ordered breadth-first
is_child_of(peter, emily).
is_child_of(peter, john).
is_child_of(laura, emily).
is_child_of(laura, john).
is_child_of(ryan, ann).
is_child_of(ryan, peter).
is_child_of(gemma, laura).
is_child_of(gemma, boris).
```

Definition of the uncle and aunt relations:

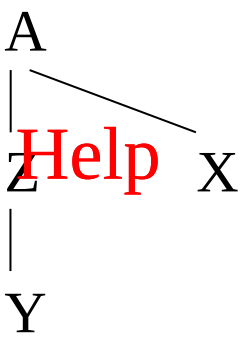
STEP 3

- body of a rule on a new line

- every subgoal on a new line with indentation (e.g. 4 whitespaces)

```
uncle(X,Y):-
        is_child_of(Y,Z),
        is_child_of(Z,A),
        is_child_of(X,A),
        X\=Z,
        male(X).

aunt(X, W) :-
        is_child_of (W,Z),
        is_child_of (Z,A),
        is_child_of (X,A),
        X\=Z,
        female(X).
```



STEP 4

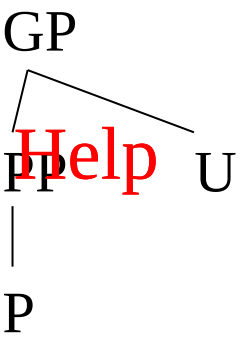
- use meaningful variable names

e.g. in the *uncle* rule: *Unc* (or *Uncle* or *U*) instead of X, *Person* (or *P*) instead of Y.

Similarly in the *aunt* rule: Use *Aun* (or *Aunt* or *A*).

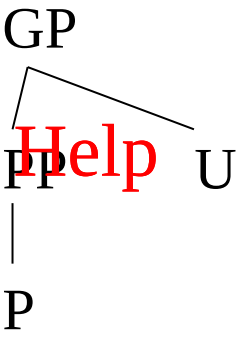
```
uncle(U,P):-
        is_child_of(P,PP),
        is_child_of(PP,GP),
        is_child_of(U,GP),
        U\=PP,
        male(U).

aunt(A, P) :-
        is_child_of(P,PP),
        is_child_of(PP,GP),
        is_child_of(A,GP),
        A\=PP,
        female(A).
```



```
uncle(U,P):-
        is_child_of(P,PP),
        is_child_of(PP,GP),
        is_child_of(U,GP),
        U\=PP,
        male(U).
```

siblings(P,U)




```
aunt(A, P) :-
        is_child_of(P,PP),
        is_child_of(PP,GP),
        is_child_of(A,GP),
        A\=PP,
        female(A).
```

siblings(PP,A)

STEP 5

- define auxiliary predicates:

e.g. siblings (+comments explaining the predicate)

```
% Child1 and Child2 are siblings if they are
% different children of the same parent.

siblings(Child1, Child2) :-
        is_child_of(Child1, Parent),
        is_child_of(Child2, Parent),
        Child1 \= Child2.
```

```
uncle(U, P) :-
        is_child_of(P, PP),
        siblings(U, PP),
        male(U).

aunt(A, P) :-
        is_child_of(P, PP),
        siblings(A, PP),
        female(A).
```

Summary: Bad Style

```
female( emily ).
female(gemma).
child(peter, john).
child(laura, emily) .
female(laura  ).female(ann).
female(lucy).
child(laura,john).
child(ryan, ann).
male(peter).

        male(boris  ).
        child( gemma, laura ).
male(  ryan).
male( john)   .
```

```
child(peter, emily ).

child(ryan        , peter).
child(gemma, boris      ).
uncle(X,Y):-child(Y,Z),child(Z,A),
child(X,A),X\=Z,male(X).
aunt(X, W) :-
child(W,Z),child(Z,A), child(X,A),
X\=Z,female(X).
```

To: Good Style

```
% all females - ordered breadth-first
female(emily).
female(ann).
female(laura).
female(lucy).
female(gemma).

% all males - ordered depth-first
male(john).
male(peter).
male(ryan).
male(boris).
```

```
\* is_child_of(Child, Parent) means that Child is the child % of Parent
*/ ordered breadth-first
is_child_of(peter, emily).
is_child_of(peter, john).
is_child_of(laura, emily).
is_child_of(laura, john).
is_child_of(ryan, ann).
is_child_of(ryan, peter).
is_child_of(gemma, laura).
is_child_of(gemma, boris).
```

To: Good Style cntd.

```
uncle(U, P) :-
        is_child_of(P, PP),
        siblings(U, PP),
        male(U).

aunt(A, P) :-
        is_child_of(P, PP),
        siblings(A, PP),
        female(A).
```



```
% Child1 and Child2 are siblings if they are different children of the same parent.
siblings(Child1, Child2) :-
        is_child_of(Child1, Parent),
        is_child_of(Child2, Parent),
        Child1 \= Child2.
```

Prolog – Good Layout Style Summary

- COMMENT your code: header, predicate-description, ...
- Use whitespaces consistently
- Each clause begins in a new line
- Rules have the form:

```
head :-
        subgoal1,
        subgoal2,
        ...
        last_subgoal.
```

- Indentation: whitespaces

Summary cntd.

- Predicate-groups: all clauses of one predicate together
- Vertical space between predicate-groups indicates "distance"
- Limit the length of a clause (i.e. the number of subgoals) by using

  auxiliary predicates.

- Choose meaningful (& pronouncible?) names for variables and predicates.
- Prolog-programmers seem to prefer using underscores:

  is_uncle_of instead of isUncleOf

- Name of a predicate should indicate the meaning of its arguments:

  mother(X,Y)

  mother_of(X,Y)

  is_mother_of(X,Y)

  mother_child(X,Y)

- Note that different predicates can have the same name if their number of arguments are different:


```
mother(X,Y)
mother(X,Y,Z)
```

But it is better if you don't do this!

- Argument order:

For example for an accumulator-style predicate

```
acc(Input,Intermediate,Output)
reverse_acc(InputList,IntermediateResult,ReversedList)
```


- Use auxiliary predicates to decrease the number of subgoals in a clause:


```
head :-
        subgoal1, subgoal2, subgoal3, subgoal4, subgoal5, subgoal6.
```

Package up some of the subgoals into an auxiliary definition.

This helps readability and re-usability.

You have some options, e.g.:

```
head :-
        subgoal1, subgoal2, aux, subgoal6.
aux:-
        subgoal3, subgoal4, subgoal5.
```

Or

```
head :-
        aux1, aux2.
aux1:-
        subgoal1, subgoal2, subgoal3.
aux2:-
        subgoal4, subgoal5, subgoal6.
```


You decide which aux definition may be more useful/re-usable.

- *Tail recursion* is efficient, but don't worry about it too much.
- TEST your program!
- Test your program incrementally as you are developing it.
- Trace. / notrace.

Useful for debugging and for understanding the Prolog query evaluation strategy.

Useful Tips and Common Mistakes

- The Sicstus Manual:

Http://sicstus.sics.se/documentation.html

Available under “help” when you invoke Sicstus.

- “Coding Guidelines for Prolog" by Covington et al. (2012), **Programming,** Theory and Practice of Logic Programming, Volume 12 / Issue 06 / November 2012, pp 889-927. The pdf is online.

Tips and Common Mistakes: usage of comma ","

- commas are only used in the body of a rule:

  head :- subgoal1, ... , last subgoal.

- You cannot separate facts by a comma:

Each fact begins on a new line and has a full stop (.) at the end.

✘ country(britain), country(holland), country(France).

✔ country(britain).

✔ country(holland).

✔ country(France).

- You cannot use commas in the head of a rule.

  ✘ wet(X), cold(X) :- raining, outside(X).

- Prolog warning:

!Permission error: cannot redefine built-in ','/2

The head of a rule is always a single atomic formula.

✔ wet(X) :- raining, outside(X).

✔ cold(X) :- raining, outside(X).

Tips and Common Mistakes: Nesting

Prolog does not allow nesting:

You cannot use

```
is_mother_of(Mother, Child) :-
        is parent_of(female(Mother), Child).
```


Correct version:

```
is_mother_of(Mother, Child) :-
        is_parent_of(Mother, Child),
        female(Mother).
```

Tips and Common Mistakes: Variables

- Remember: Variables start in the upper case and anything starting with an upper case letter is a variable.
- Think carefully before you use variables in the heads of condition-less clauses!

E.g. If you specify

person(X).

Logically you have specified

∀X person(X),

and your program will say “yes”, for example to a query such as

| ?- person(logic_course).

- Variables are normally used to express dependencies:

```
is_mother_of(Mother, Child) :-
        is_child_of(Child, Mother),
        female(Mother).
```


- If one of the variables doesn't matter for the dependencies, you can use an anonymous variable, i.e. underscore "_".
- If "_" appears multiple times in the same clause, the occurrences refer to *distinct* variables.

Tips and Common Mistakes: Singleton Variables

- A very common Prolog warning:

[…, …] - singleton variables

Example:

```
parent(P) :-
        is_child_of(Child, P).
```

[Child] - singleton variables

- This is a warning to help you with two common mistakes:
  - Spelling mistakes in variables
  - Forgetting to use/bind a variable
- It indicates that there is one or more variable in the clause that appears only once.

Tips and Common Mistakes: Another Common Warning

Existence error in user: …..

E.g. parent(P) :- child_of(Child, P).

Query: ?- parent(X).

! Existence error in user:child_of/2! procedure user:child_of/2 does not exist! goal: user:child_of(_128,_129)

Prolog is expecting to find a definition for child_of/2, but cannot find it.

You may have forgotten to define it, or you may have defined it but

- you have used a wrong number of arguments, or
- you have a spelling mistake, e.g. childOf or is_child_of instead of child_of.

Tips and Common Mistakes: Disjunctions

- Disjunction has to be used with parentheses:

  subgoal1 $\wedge$ (subgoal2 $\vee$ subgoal3) becomes

  subgoal1, (subgoal2; subgoal3)

- Some people also prefer this presentation:

  subgoal1,

  (subgoal2

  ; subgoal3

)

Tips and Common Mistakes: is

The "is" predicate:

- Used to evaluate arithmetic expressions.
- LHS is a variable or a constant, RHS should be a ground expression when the predicate is called.

Example

```
| ?- X=5, Y is X+3 ; Y is X+5.
X = 5,
Y = 8 ? ;
! Instantiation error in argument 2 of is/2! goal:  _116 is
_119+5
| ?- X=5, (Y is X+3 ; Y is X+5).
X = 5,
Y = 8 ? ;
X = 5,
Y = 10 ? ;
no
```

Tips and Common Mistakes: Others

Order matters:

- In recursive definitions:
  - Base case first
  - Then the recursive clause
- Order of subgoals matters too.

Good info about Debugging, spy, etc

https://sicstus.sics.se/sicstus/docs/3.7.1/html/sicstus_9.html