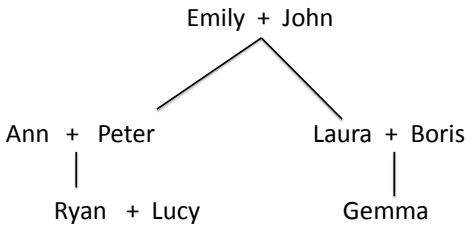
Good vs Bad style of Presenting Prolog Programs

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

```
% % by Claudia Schulz
% 11th November 2013
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% STEP 1
% - consistent use of whitespaces
% - all clauses of one predicate together
% - different predicates are separated by spaces
% - every clause begins in a new line

% STEP 2
% - comments explain the predicates
% - 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):-
        child(Y,Z),
        child(Z,A),
        child(X,A),
        X\=Z,
        male(X).
aunt(X, Y) :-
        child(Y,Z),
        child(Z,A),
        child(X,A),
        X\=Z,
        female(X).
```

% STEP 4

% - use meaningful variable names in rule 1 (X = Uncle, Y = Person). Similarly in rule 2

% STEP 5

% - define auxiliary predicates: siblings (+comments)

```
uncle(Uncle, Person) :-
        is_child_of(Person, Persons_Parent),
        siblings(Uncle, Persons_Parent),
        male(Uncle).

aunt(Aunt, Person) :-
        is_child_of(Person, Persons_Parent),
        siblings(Aunt, Persons_Parent),
        female(Aunt).
```

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

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

So: 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(Uncle, Person) :-
        is_child_of(Person, Persons_Parent),
        siblings(Uncle, Persons_Parent),
        male(Uncle).

aunt(Aunt, Person) :-
        is_child_of(Person, Persons_Parent),
        siblings(Aunt, Persons_Parent),
        female(Aunt).

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

Prolog – Good Layout Style

- 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 (e.g. 4)

- 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.

- 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
)
```

- Choose meaningful (& pronouncable?) 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: predicate(Input,Intermediate,Output)
  reverse list(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.

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

- Tail recursion if efficient, but don't worry about it too much
- TEST your program!

Useful Tips and Common Mistakes

- The Sicstus Manual:
  Http://sicstus.sics.se/documentation.html

- Coding Guidelines for Prolog" by Covington et al. (2012)

- Trace. / notrace.

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

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 dot (.) at the end.

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

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

- Prolog warning:

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

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 the 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) :-
        child_of(Child, P).
```

[Child] - singleton variables

- This is a warning to help you with two common mistakes:
  - Spelling mistakes in variables
  - Forget 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 instead of child_of.

Tips and Common Mistakes: Others

Order matters:

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

The "is" predicate:

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

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).
```