Practical "Introduction to Artificial Intelligence"

Prof. Dr. Gunter Grieser

Block 1: Prolog

Sheet 2: Unification Search and Recursion

Hints:

- *In Block 1 (Prolog) you do not have to submit your solutions to me. Just solve the excercies and discuss your problems and solutions.The aim of Block 1 is that you become familiar with the prolog programming.*
- *If you do not succed with a task, just delay it and try it again later. Some constructs need time to settle in the brain and will become easier as you get more experienced.*

Preparation:

Read Chapters 2 and 3 of LearnPrologNow!.

Excercise 2.1

Reproduce the examples from the two chapters of LearnPrologNow! on your machine and solve the excercises.

Excercise 2.2

Which of the following pairs of terms can be unified? How are the variables instantiated in these cases?

- `donut` and `croissant`
- `animal(X)` and `animal(toto)`
- `Food` and `'food'`
- `tasty` and `'tasty'`
- `good(pizza, Y)` and `good(X, salami)`
- `f(a, X, Y)` and `f(X, Y, b)`
- `plus(sqr(a), X)` and `plus(sqr(Y), mult(b, Y))`

Excercise 2.3

A dance school asks for your assistance: They invented an electronic dance course and hired you to program certain prolog modules.

- A dance is a sequence of possible moves (steps and turns) ended by 'stop'.
  - steps[1]: 'left', 'right', 'forward', 'backward'
  - 360° turns: 'left turn', 'right turn'
- A dance is modeled as a prolog term describing the sequence of steps.
  - e.g. `left(forward(right(backward(stop))))` models a dance called "square".

a) Write a predicate `correct_dance/1` that is true iff the argument is a prolog term that describes a dance.

- E.g. `correct_dance(left(right(stop)))` shall be true whereas `correct_dance(f(1,2))` or `correct_dance(left)` are not.

b) A dance is called symmetric, if

- every step is followed by its corresponding counterpart.
  - E.g. a step to the left is followed by a step to the right.
- Every turn is followed by another turn, however, the direction here doesn't matter.

Write a predicate `symmetric_dance/1` that is true iff the argument is a prolog term that describes a symmetric dance.

c) If a couple dances together, they should avoid kicking each others feets. Write a predicate `safe_dance(D1,D2)` that verifies that the dances `D1` and `D2` of the two partners are safe. This is the case if:

- If both partners make a step, they move to the same direction (e.g. both to the left)
- The partners never turn at the same time.
- Both partners end at the same moment.

d) (tricky) Write a predicate `idempotent_dance/1` that is true iff the dancer ends at the same location as it started.

e) Invent a new dance (for singles or couples) and write a predicate in prolog that checks whether the dance is correct.

1 The dancer always keeps its viewing direction. I.e. after a step to the left, the dancer still looks in the same direction, but has moved one step to the left.

Excercise 2.4

Consider the following program:

```
diff(x,1).
diff(C, 0):-
     atomic(C),
     C \= x.
diff(F + G, DF + DG) :-
     diff(F, DF),
     diff(G, DG).
diff(F - G, DF - DG) :-
     diff(F, DF),
     diff(G, DG).
diff(F * G, DF * G + F * DG) :-
     diff(F, DF),
     diff(G, DG).
diff(F / G, (DF * G - F * DG) / (G * G)) :-
     diff(F, DF),
     diff(G, DG).
```

a) Analyse the program:

- Which result is this program computing?
- Try this program for different values, trace it and understand the tracing mechanism.

b) Extend the program for other functions, e.g. *sin(x)* or x^n.

c) Extend the program such that the name for the "variable" (in the example: *x*) can be provided as a parameter.

d) Extend your program from c) such that two variables can be given.

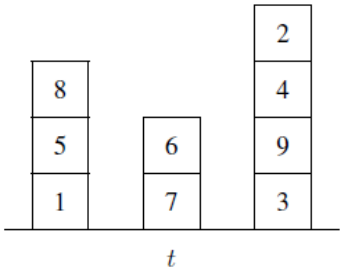
Excercise 2.5

Write a program analogous to the previous excercise that simplifies terms, e.g.

- 0 + x = x, x + x = 2 * x
- 0 * x = 0, 1 * x = x
- x ^ 0 = 1, x ^1 = x

Excercise 2.6

At a table *t* there are numbered blocks stacked as follows.



a) Write a prolog program that models the depicted situation by a predicate `stands_on/2`. Here, `stands_on(X,Y)` means that block `X` stands on block `Y`.

b) Define a predicate `above/2` using `stands_on/2` that models that a block is above another block or the table.

c) Draw the search trees for the following queries. Issue the queries, trace the computation and compare it to your search tree:

- Does block 8 stands on block 1?
- Is block 2 above block 7?
- What is over block 3?
- What is below block 4?