Practical "Introduction to Artificial Intelligence"

Prof. Dr. Gunter Grieser

Block 1: Prolog

Sheet 3: Lists and Arithmetics

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 (at home):

Read Chapter 4 and 5 of LearnPrologNow!.

Excercise 3.1

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

Excercise 3.2

What are the answers to the following queries? First answer the question, then try it with prolog.

- `?- [a, X, a] = [Y, b, Y].`
- `?- [Y, c] = [c, Y | []].`
- `?- [_, [] | [a, Y]] = [a, _, Z, b].`
- `?- [a | [b | T]] = [X, H | [c | [d]]].`

Excercise 3.3

The following family connections are modeled in prolog as `family(Name, Father, Mother, List_of_Children)`:

```
family(meier, uwe, erika, [birgit, hans, elke]).
family(hoffmann, werner, maria, []).
family(mueller, heinz, monika, [peter, susanne]).
```

```
family(schulz, karl, claudia, [max, fritz, karl, grete]).
```

a) Consider the following queries and reproduce the process of solution findig:
- What are the prenames of the parents with children Birgit, Hans and Elke?
- What are the children of family Mueller?
- Which families have no children?

b) Which families are covered by the following queries?
- `?- family(Name,_,_,[_,_]).`
- `?- family(Name,_,_,[]).`

c) Identify the names of families with
- exactly 3 children.
- at least 1 child.
- at least 2 children.

d) Identify the names of families with at most 2 children.

Excercise 3.4

The predicate member/2 is defined as follows (see Ch. 4.2 of LearnPrologNow!):

```
member(X,[X|T]).
member(X,[H|T])  :-  member(X,T).
```

Draw the search tree of `?- member(a, [b, X, a]).`

Excercise 3.5

Define a predicate `last(L,E)` that is true if `E` is the last element of list `L`.

Excercise 3.6

Define a predicate `increase(List1, List2)` is true if `List2` is a list where all elements of `List1` are increased by 1. (We assume that `List1` only contains integers).

Excercise 3.7

Define a predicate `delete(List1, X, List2)` that is true if `List2` is a list that results from deleting all occourences of element `E` from `List1`.

Excercise 3.8

Define a predicate `swap(List1, List2)` that swaps the two neighbouring elements thouroughly.

- Example: `swap([1,2,3,4], [2,1,4,3]).`
- Tip: first consider lists of even length. After that extend your program arbitrary lists.

Excercise 3.9

Define a predicate `list_length(List,Length)` that determines the length of a list.

- What happens if you query `list_length(List,3)`?

Excercise 3.10

Define a predicate `replace_element(List1, X, Y, List2)` that is true if `List2` is a list that results from replacing all occourences of element `X` by `Y` in `List1`.

Excercise 3.11

Define a predicate `listsum(List, Sum)` that sums up all elements of the list of integers.

Excercise 3.12

Define a predicate `first_n(N, List)` that generates a list of the first N natural numbers (1...N).

Change your program so that it generates the numbers in inverse order (N...1).