Hochschule für Technik Stuttgart

1: Append in Prolog II

Call:acc_append(_4934,_4938,[1, 2])
Exit:acc_append([],[1, 2],[1, 2])
X = [],
Y = [1, 2]

In Prolog, one list (written as [a,b,c] or [] for the empty list) is appended to another by the following code:

```
acc_append([], Ys, Ys).
acc_append([X|Xs], Ys, [X|Zs]) :- acc_append(Xs, Ys, Zs).
```

What happens if you call acc_append „in reverse" – giving variables for the two input lists and a result list for the accumulator/result variable? Why? Trace the Prolog query and show the search tree for acc_append(X, Y, [1,2]).

2: List-reverse in Prolog

Here is the LISP code for list reverse using an accumulator. Re-write it in Prolog.

```
(define list-reverse-aux (lambda (L A)
            (if (null? L) A
                (list-reverse-aux (cdr L) (cons (car L) A)))))
(define list-reverse (lambda (L) (list-reverse-aux L '())))
```

trace,list_reverse([1,2,3], [], 0)

```
list_reverse([],A,A).
list_reverse([H|T], A, O):- list_reverse(T, [H|A], O)
```

[A|H] will fail if written here

O cannot be empty list here

Hint 1: There is no cons predicate; use the [Head|Rest] notation to create a new list.

Hint 2: If you want to see the result of the list reversal, your query will have to contain an unbound variable that takes on the value of A eventually (in addition to the actual accumulator).

```
list_reverse([],A,A).
list_reverse([H|T], A, O):- list_reverse(T, [H|A], O)
```

trace,list_reverse(X, [], [3,2,1])

How many ways to arrive ? 1

3: CFGs and DCGs

Expand the sample DCG to cover the following grammar (from Lecture "Syntax and Semantics", slide 5):

- Expr ==> Term | Term '+' Expr
  Term ==> Factor | Factor '*' Term
  Factor ==> '0' | ... | '9' | '(' Expr ')

```
expr --> term, ['+'], term.
term --> id.

id --> [1].
id --> [2].
```

To get results: expr([2, '+', 2, '-', 3], []).