

# Computational Linguistics (INF2820 — More Lisp)

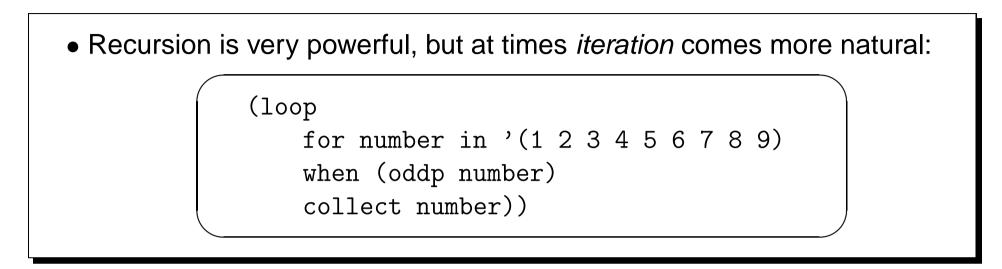
(defun ? (n) (if (equal n 0) 1 (\* n (! (- n 1)))))

**Stephan Oepen** 

Universitetet i Oslo

oe@ifi.uio.no

## **Iteration — Another Control Structure**



#### Some loop() Directives

- for symbol { in | on } list iterate symbol through list elements or tails;
- for symbol from start [to end] [by step] count symbol in range;
- [{when|unless} test] {collect|append} sexp accumulate sexp;
- [while test] do sexp<sup>+</sup> execute expression(s) sexp<sup>+</sup> in each iteration.



# **A Few More Examples**

- loop() is extremely general; a single iteration construct fits all needs: ? (loop for foo in '(1 2 3) collect foo)  $\rightarrow$  (1 2 3) ? (loop for foo on '(1 2 3) collect foo)  $\rightarrow$  ((1 2 3) (2 3) (3)) ? (loop for foo on '(1 2 3) append foo)  $\rightarrow$  (1 2 3 2 3 3) ? (loop for i from 1 to 3 by 1 collect i)  $\rightarrow$  (1 2 3)
  - loop() returns the final value of the accumulator (collect or append);
  - return() terminates the iteration immediately and returns a value:

```
? (loop for foo in '(1 2 3) when (evenp foo) do (return foo)) \rightarrow 2
```



# Input and Output — Side Effects

- Input and output, to files or the terminal, is mediated through streams;
- the symbol t can be used to refer to the default stream, the terminal:

```
? (format t "line: ~a; token '~a'.~%" 42 "foo") \sim line: 42; token 'foo'.
\rightarrow nil
```

- (read stream nil) reads one well-formed s-expression from stream;
- (read-line *stream* nil) reads one line of text, returning it as a string;
- the second argument to reader functions asks to return nil on end-of-file.

```
(with-open-file (stream "sample.txt" :direction :input)
  (loop
    for line = (read-line stream nil)
    while (not (null line)) do (format t "~a~%" line)))
```



- INF2820 — 12-FEB-10 (oe@ifi.uio.no)

## **Fine Points of Strings and Regular Expressions**

- Need to escape double quote (") in strings, e.g. "foo \"bar\" baz";
- likewise for RE operators, to force literal match, e.g. /\([a-z]+\)\./;
- backslash is escape character for Lisp strings  $\rightarrow$  "\\([a-z]+\\)\\.";
- REs in Lisp represented as strings, thus need two levels of escaping.
- The Portable Perl-Compatible Regular Expressions package for Lisp;

• many more functions in PPCRE library; see the on-line documentation.



INF2820 - 12-FEB-10 (oe@ifi.uio.no)