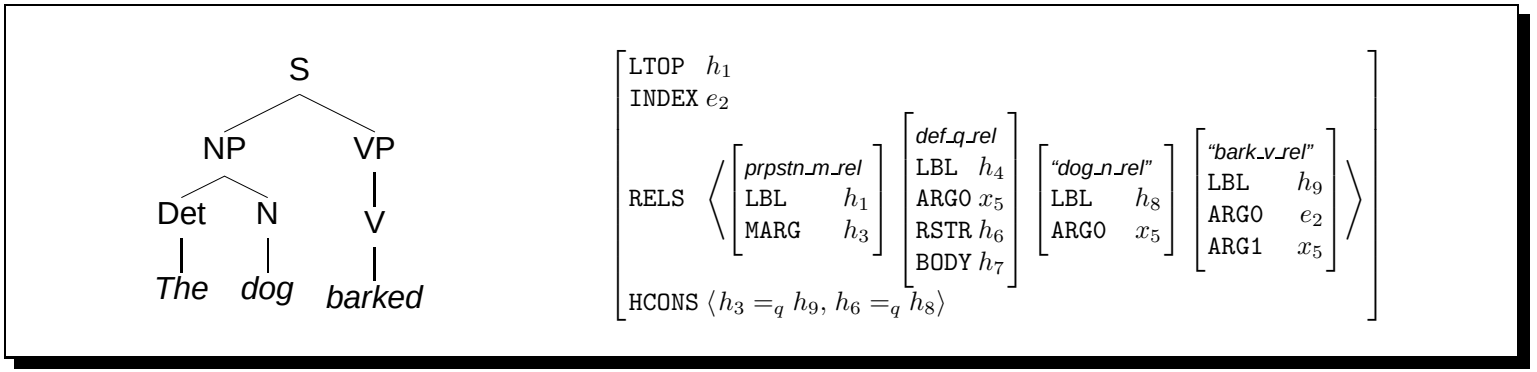

Computational Linguistics (INF2820 — Data Abstraction)

(defun ? (n) (if (= n 0) 1 (* n (! (- n 1)))))

Stephan Oepen

Universitetet i Oslo & CSLI Stanford

oe@ifi.uio.no

Vectors and Arrays

- Multidimensional 'grids' of data can be represented as *vectors* or *arrays*;
- `(make-array (`*rank*$_1$ ... *rank*$_n$`))` creates an array with *n* dimensions;

```
? (setf *foo* (make-array '(2 5) :initial-element 0))
→ #((0 0 0 0 0) (0 0 0 0 0))
? (setf (aref *foo* 1 2) 42) → 42
```

	0	1	2	3	4
0	0	0	0	0	0
1	0	0	42	0	0

- all dimensions count from zero; `aref()` accesses one individual cell;
- one-dimensional arrays are called *vectors* (abstractly similar to lists).

Abstract Data Types

- `defstruct()` creates a new *abstract data type*, encapsulating a structure:

```
? (defstruct rule
    lhs rhs)
→ RULE
```

- `defstruct()` defines a new *constructor*, *accessors*, and a type *predicate*:

```
? (setf *foo* (make-rule :lhs 'S :rhs '(NP PP)))
→ #S(RULE :LHS S :RHS (NP PP))
? (listp *foo*) → nil
? (rule-p *foo*) → t
? (setf (rule-rhs *foo*) '(NP VP)) → (NP VP)
? *foo* → #S(RULE :LHS S :RHS (NP VP))
```

- abstract data types *encapsulate* a group of related data (i.e. an 'object').