Konkrete Syntax

- **Lexikalische Syntax:** Darstellung von Wortfolgen durch Zeichenfolgen
- **Phrasale Syntax:** Darstellung von Bäumen durch Wortfolgen

- **Lexer:** Übersetzung Zeichenfolge $\longrightarrow$ Wortfolge
  (falls Zeichenfolge lexikalisch zulässig)
- **Parser:** Übersetzung Wortfolge $\longrightarrow$ Baum
  (falls Wortfolge Satz gemäß der syntaktischen Kategorie)

Ein Lexer für Typen von F

```
exception Error of string

datatype token = BOOL | INT | ARROW | LPAR | RPAR

fun lex nil = nil
  | lex (#" ":: cr) = lex cr
  | lex (#"\t":: cr) = lex cr
  | lex (#"\n":: cr) = lex cr
  | lex (#"b":: #"o":: #"o":: #"l"::cr) = BOOL:: lex cr
  | lex (#"i":: #"n":: #"t"::cr) = INT:: lex cr
  | lex (#"-":: #">"::cr) = ARROW :: lex cr
  | lex (#"("::cr) = LPAR :: lex cr
  | lex (#")"::cr) = RPAR :: lex cr
  | lex t = raise Error ("cannot lex: "^implode t)

val t1 = lex (explode "(int->bool)->int")
val t2 = lex (explode "   intbool->int   ")
val t3 = lex (explode "intboolboo")
```

Phrasale Syntax

- **Phrasale Syntax für Typen**

```
ty := pty | pty "->" ty
pty ::= "bool" | "int" | "(" ty ")"
```

- **Affinität:** Eine konkrete Grammatik ist affin zu einer abstrakten Grammatik wenn
  - jede konkrete Ableitung durch genau eine abstrakte Ableitung simuliert werden kann
  - jede abstrakte Ableitung durch mindestens eine konkrete Ableitung simuliert werden kann

- Eine konkrete Grammatik heißt **eindeutig**, wenn es zu einem Satz und einer Kategorie höchstens einen Syntaxbaum gibt, der den Satz aus der Kategorie ableitet.

- Affinität + Eindeutigkeit $\Rightarrow$ Zulässige Wortdarstellungen beschreiben immer genau eine Phrase.

Abstrakte Syntax von *F*

$$
\begin{array}{l}
z \in \mathbb{Z} \\
c \in \mathit{Con} = \mathit{false} \mid \mathit{true} \mid z \\
x \in \mathit{Id} = \mathbb{N} \\
t \in \mathit{Ty} = \mathit{bool} \mid \mathit{int} \mid t \rightarrow t \\
\\
o \in \mathit{Opr} = + \mid - \mid * \mid \leq \\
\\
e \in \mathit{Expr} = \\
\qquad c \\
\qquad \mid x \\
\qquad \mid e\ o\ e \\
\qquad \mid \mathit{if}\ e\ \mathit{then}\ e\ \mathit{else}\ e \\
\qquad \mid \mathit{fn}\ x\ :\ t\ \Rightarrow e \\
\qquad \mid e\ e
\end{array}
$$

```
datatype con = False | True
     | IC of int
type id = string
datatype ty = Bool | Int
     | Arrow of ty * ty
datatype opr = Add | Sub
     | Mul | Leq
datatype exp =
        Con of con
      | Id of id
      | Opr of opr * exp * exp
      | If of exp * exp * exp
      | Abs of id * ty * exp
      | App of exp * exp
```