

# 1 Lexikalische Syntax

Vorgegeben sind

- die Menge  $Int$  aller (Darstellungen von) ganzen Zahlen  $n$ ,
- die Menge  $Bool = \{true, false\}$  der booleschen Werte  $b$ ,
- die Menge  $Unit = \{()\}$ , die nur ein einziges Element  $()$  enthält,
- eine unendliche Menge  $Id$  von *Namen* (engl.: *identifiers*)  $id$ .

# 2 Kontextfreie Syntax

Wir definieren

- die Menge  $Op$  aller *Operatoren* (engl.: *operators*)  $op$
- die Menge  $Const$  aller *Konstanten* (engl.: *constants*)  $c$
- die Menge  $BType$  aller *Basistypen* (engl.: *base types*)  $\beta$
- die Menge  $Type$  aller *Typen* (engl.: *types*)  $\tau$
- die Menge  $Exp$  aller *Ausdrücke* (engl.: *expressions*)  $e$
- und die Menge  $Dec$  aller *Deklarationen* (engl.: *declarations*)  $d$

durch die folgende kontextfreie Grammatik

$op ::=$	$\sim$	unäres Minus
	$  \text{ not}$	Negation
	$  + \mid - \mid * \mid \mathbf{div} \mid \mathbf{mod}$	binäre arithmetische Operatoren
	$  < \mid > \mid \leq \mid \geq$	Vergleichsoperatoren
	$  =$	Gleichheit
	$  \#n \quad (n > 0)$	Projektionen
$c ::=$	$()$	unit-Element
	$  b$	boolescher Wert
	$  n$	ganze Zahl
	$  op$	Operator
$\beta ::=$	$\mathbf{unit} \mid \mathbf{bool} \mid \mathbf{int}$	
$\tau ::=$	$\beta$	Basistyp
	$  \tau_1 * \dots * \tau_k$	( $k \geq 2$ ) Produkttyp
	$  \tau_1 \rightarrow \tau_2$	Funktionstyp
$e ::=$	$c$	Konstante
	$  id$	Name
	$  (e_1, \dots, e_k)$	( $k \geq 2$ ) Tupel
	$  e_1 e_2$	Applikation
	$  \mathbf{if} \ e_0 \ \mathbf{then} \ e_1 \ \mathbf{else} \ e_2$	bedingter Ausdruck
	$  \mathbf{let} \ d \ \mathbf{in} \ e_1 \ \mathbf{end}$	let-Ausdruck
	$  \lambda id : \tau. e_1$	$\lambda$ -Abstraktion
$d ::=$	$\mathbf{val} \ id = e$	

## 3 Typsystem

### 3.1 Typurteile für Konstanten

Ein *Typurteil für Konstanten* (engl.: *typing judgment*) ist eine "Formel" der Gestalt  $c :: \tau$ . Die *gültigen* Typurteile sind durch folgende Axiome festgelegt.

- (UNIT)  $() :: \mathbf{unit}$
- (BOOL)  $b :: \mathbf{bool}$
- (INT)  $n :: \mathbf{int}$
- (UMINUS)  $\sim :: \mathbf{int} \rightarrow \mathbf{int}$
- (NOT)  $\mathit{not} :: \mathbf{bool} \rightarrow \mathbf{bool}$
- (AOP)  $op :: \mathbf{int} * \mathbf{int} \rightarrow \mathbf{int}$  falls  $op \in \{+, -, *, \mathbf{div}, \mathbf{mod}\}$
- (ROP)  $op :: \mathbf{int} * \mathbf{int} \rightarrow \mathbf{bool}$  falls  $op \in \{<, >, \leq, \geq\}$
- (EQ)  $= :: \beta * \beta \rightarrow \beta$
- (PROJ)  $\#n :: \tau_1 * \dots * \tau_k \rightarrow \tau_n$  falls  $1 \leq n \leq k$

### 3.2 Typurteile für Ausdrücke

**Definition:** Eine *Typumgebung* (engl.: *type environment, type assignment*) ist eine endliche partielle Funktion  $\Gamma : Id \hookrightarrow Type$ , d.h. eine partielle Funktion, deren Definitionsbereich (engl.: *domain*)  $dom(\Gamma)$  endlich ist.

Ein *Typurteil für Ausdrücke* ist von der Form  $\Gamma \triangleright e :: \tau$ . Die *gültigen* Typurteile erhalten wir mit den folgen Regeln.

- (CONST) 
$$\frac{c :: \tau}{\Gamma \triangleright c :: \tau}$$
- (ID)  $\Gamma \triangleright id :: \tau$  falls  $id \in dom(\Gamma)$  und  $\Gamma(id) = \tau$
- (TUPLE) 
$$\frac{\Gamma \triangleright e_1 :: \tau_1 \quad \dots \quad \Gamma \triangleright e_k :: \tau_k}{\Gamma \triangleright (e_1, \dots, e_k) :: \tau_1 * \dots * \tau_k}$$
- (APP) 
$$\frac{\Gamma \triangleright e_1 :: \tau \rightarrow \tau' \quad \Gamma \triangleright e_2 :: \tau}{\Gamma \triangleright e_1 e_2 :: \tau'}$$
- (COND) 
$$\frac{\Gamma \triangleright e_0 :: \mathbf{bool} \quad \Gamma \triangleright e_1 :: \tau \quad \Gamma \triangleright e_2 :: \tau}{\Gamma \triangleright \mathbf{if} \ e_0 \ \mathbf{then} \ e_1 \ \mathbf{else} \ e_2 :: \tau}$$
- (LET-VAL) 
$$\frac{\Gamma \triangleright e_1 :: \tau_1 \quad \Gamma[\tau_1/id] \triangleright e_2 :: \tau_2}{\Gamma \triangleright \mathbf{let} \ \mathbf{val} \ id = e_1 \ \mathbf{in} \ e_2 \ \mathbf{end} :: \tau_2}$$
- (ABSTR) 
$$\frac{\Gamma[\tau/id] \triangleright e :: \tau'}{\Gamma \triangleright \lambda id : \tau. e :: \tau \rightarrow \tau'}$$