

Abstrakte Syntax der Sprachen \mathcal{L}_0 und \mathcal{L}_1

Vorgegeben seien

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

Definiert werden

- die Menge Op der binären *Operatoren* (engl.: *operators*) op ,
- die Menge $Const$ der *Konstanten* (engl.: *constants*) c ,
- die Menge Exp aller *Ausdrücke* (engl.: *expressions*) e

durch die folgende KFG

$op ::= + \mid - \mid * \mid / \mid \text{mod}$	arithmetische Operatoren
$\mid < \mid > \mid \leq \mid \geq \mid =$	Vergleichsoperatoren
$c ::= ()$	unit-Element
$\mid b$	boolescher Wert
$\mid n$	ganze Zahl
$\mid op$	Operator
$e ::= c$	Konstante
$\mid id$	Name
$\mid e_1 e_2$	Applikation
$\mid \text{if } e_0 \text{ then } e_1 \text{ else } e_2$	bedingter Ausdruck
$\mid \lambda id. e_1$	λ -Abstraktion
$\mid \text{let } id = e_1 \text{ in } e_2$	let-Ausdruck

Die so definierte Sprache bezeichnen wir mit \mathcal{L}_1 . Die Sprache $\mathcal{L}_0 \subseteq \mathcal{L}_1$ sei definiert durch

$e ::= id$	Name
$\mid e_1 e_2$	Applikation
$\mid \lambda id. e_1$	λ -Abstraktion

\mathcal{L}_0 heißt *reiner ungetypter λ -Kalkül* (engl.: *pure untyped λ -calculus*). \mathcal{L}_1 ist ein *angewandter ungetypter λ -Kalkül* (engl.: *applied untyped λ -calculus*).

Konkrete Syntax

Die Applikation hat höchste Priorität, d.h. sie bindet stärker als λ , **let** **in** und **if** **then** **else** $_$, und sie ist linksassoziativ, d.h. $e_1 e_2 e_3$ steht für $(e_1 e_2) e_3$.