Formelsammlung Theorie der Programmierung

Inhaltsverzeichnis

1. Kontextfreie Grammatik

1.1. Operationen

$$
\begin{aligned}
\mathrm{op} \; ::= \; & + \mid - \mid * \mid / \mid \mathrm{mod} \\
& \mid < \mid > \mid \leqslant \mid \geqslant \mid = \\
& \mid :=
\end{aligned}
$$

1.2. Konstanten

$$
\begin{aligned}
\mathrm{c} \; ::= \; & () \mid \mathrm{b} \mid \mathrm{n} \\
& \mid \mathrm{op} \\
& \mid [\,] \mid \mathrm{cons} \mid \mathrm{hd} \mid \mathrm{tl} \mid \mathrm{is\_empty} \\
& \mid \mathrm{fst} \mid \mathrm{snd} \\
& \mid\, ! \mid \mathrm{ref} \mid \mathrm{X}
\end{aligned}
$$

1.3. Werte

$$
\begin{aligned}
\mathrm{v} \; ::= \; & \mathrm{c} \\
& \mid \mathrm{id} \\
& \mid \mathrm{op}\,\mathrm{n} \\
& \mid \lambda\,\mathrm{id.e} \\
& \mid (\mathrm{v}_1, \mathrm{v}_2) \\
& \mid \mathrm{cons}\ \mathrm{v} \\
& \mid \textbf{object method}\ \mathrm{id}_1 = \mathrm{v}_1 \ldots \textbf{method}\ \mathrm{id}_n = \mathrm{v}_n\ \textbf{end}
\end{aligned}
$$

1.4. Ausdrücke

$$
\begin{aligned}
\mathrm{e} \; ::= \; & \mathrm{c} \\
& \mid \mathrm{id} \\
& \mid \mathrm{e}_1\,\mathrm{e}_2 \\
& \mid \textbf{if}\ \mathrm{e}_0\ \textbf{then}\ \mathrm{e}_1\ \textbf{else}\ \mathrm{e}_2 \\
& \mid \lambda\,\mathrm{id.e} \\
& \mid \textbf{let}\ \mathrm{id} = \mathrm{e}_1\ \textbf{in}\ \mathrm{e}_2 \\
& \mid \textbf{rec}\ \mathrm{id.e} \\
& \mid (\mathrm{e}_1, \mathrm{e}_2) \\
& \mid \mathrm{e}_1 ; \mathrm{e}_2 \quad \text{steht für} \quad \textbf{let}\ \mathrm{id} = \mathrm{e}_1\ \textbf{in}\ \mathrm{e}_2\ \textit{wobei}\ id \notin \mathit{free}(e_2) \\
& \mid \textbf{if}\ \mathrm{e}_0\ \textbf{then}\ \mathrm{e}_1 \quad \text{steht für} \quad \textbf{if}\ \mathrm{e}_0\ \textbf{then}\ \mathrm{e}_1\ \textbf{else}\ () \\
& \mid \textbf{while}\ \mathrm{e}_0\ \textbf{do}\ \mathrm{e}_1 \quad \text{steht für} \quad \textbf{rec}\ \mathrm{id}.\,\textbf{if}\ \mathrm{e}_0\ \textbf{then}(\mathrm{e}_1 ; \mathrm{id}) \\
& \mid \textbf{object method}\ \mathrm{id}_1 = \mathrm{e}_1 \ldots \textbf{method}\ \mathrm{id}_n = \mathrm{e}_n\ \textbf{end} \\
& \mid \mathrm{e}\#\mathrm{id}
\end{aligned}
$$

1.5. Typen

$$
\begin{aligned}
\tau \; ::= \; & \textbf{unit} \mid \textbf{bool} \mid \textbf{int} \\
\mid \; & \tau_1 \rightarrow \tau_2 \\
\mid \; & \alpha \\
\mid \; & \tau_1\, \mathrm{ref} \\
\mid \; & < \mathrm{id}_1 : \tau_1 ; \ldots ; \mathrm{id}_n : \tau_n >
\end{aligned}
$$

2. Small step Semantik

2.1. Operationen

(OP) $\text{op}\ n_1\ n_2 \to \text{op}^I(n_1, n_2)$

2.2. Beta Value

(BETA-V) $(\lambda \text{id.e})\ v \to e[v/\text{id}]$

2.3. Applikation

(APP-LEFT) $\dfrac{e_1 \to e_1'}{e_1\, e_2 \to e_1'\, e_2}$

(APP-LEFT-EXN) $\dfrac{e_1 \to \text{exn}}{e_1\, e_2 \to \text{exn}}$

(APP-RIGHT) $\dfrac{e \to e'}{v\ e \to v\ e'}$

(APP-RIGHT-EXN) $\dfrac{e \to \text{exn}}{v\ e \to \text{exn}}$

2.4. Bedingungen

(COND-EVAL) $\dfrac{e_0 \to e_0'}{\textbf{if}\ e_0\ \textbf{then}\ e_1\ \textbf{else}\ e_2 \to \textbf{if}\ e_0'\ \textbf{then}\ e_1\ \textbf{else}\ e_2}$

(COND-EVAL-EXN) $\dfrac{e_0 \to \text{exn}}{\textbf{if}\ e_0\ \textbf{then}\ e_1\ \textbf{else}\ e_2 \to \text{exn}}$

(COND-TRUE) $\textbf{if}\ \text{true}\ \textbf{then}\ e_1\ \textbf{else}\ e_2 \to e_1$

(COND-FALSE) $\textbf{if}\ \text{false}\ \textbf{then}\ e_1\ \textbf{else}\ e_2 \to e_2$

2.5. Let

(LET-EVAL) $\dfrac{e_1 \to e_1'}{\textbf{let}\ \text{id} = e_1\ \textbf{in}\ e_2 \to \textbf{let}\ \text{id} = e_1'\ \textbf{in}\ e_2}$

(LET-EVAL-EXN) $\dfrac{e_1 \to \text{exn}}{\textbf{let}\ \text{id} = e_1\ \textbf{in}\ e_2 \to \text{exn}}$

(LET-EXEC) $\textbf{let}\ \text{id} = v\ \textbf{in}\ e \to e[v/\text{id}]$

2.6. Rekursion

(UNFOLD) $\textbf{rec}\ \text{id.e} \to e[\textbf{rec}\ \text{id.e}/\text{id}]$

2.7. Listen

(HD)	$\text{hd } (\text{cons}(v_1, v_2)) \rightarrow v_1$
(HD-EXN)	$\text{hd } [\,] \rightarrow \text{empty\_list\_exn}$
(TL)	$\text{tl } (\text{cons}(v_1, v_2)) \rightarrow v_2$
(TL-EXN)	$\text{tl } [\,] \rightarrow \text{emptylist}$
(IS-EMPTY-TRUE)	$\text{is\_empty } [\,] \rightarrow \text{true}$
(IS-EMPTY-FALSE)	$\text{is\_empty } (\text{cons}(v_1, v_2)) \rightarrow \text{false}$

2.8. Paare

(FST)	$\text{fst } (v_1, v_2) \rightarrow v_1$
(SND)	$\text{snd } (v_1, v_2) \rightarrow v_2$
(PAIR-LEFT)	$\dfrac{e_1 \rightarrow e_1'}{(e_1, e_2) \rightarrow (e_1', e_2)}$
(PAIR-RIGHT)	$\dfrac{e \rightarrow e'}{(v, e) \rightarrow (v, e')}$

2.9. Boolsche Ausdrücke

2.9.1. And

(AND-EVAL)	$\dfrac{e_1 \rightarrow e_1'}{e_1 \mathbin{\&\&} e_2 \rightarrow e_1' \mathbin{\&\&} e_2}$
(AND-TRUE)	$\text{true} \mathbin{\&\&} e \rightarrow e$
(AND-FALSE)	$\text{false} \mathbin{\&\&} e \rightarrow \text{false}$

2.9.2. Or

(OR-EVAL)	$\dfrac{e_1 \rightarrow e_1'}{e_1 \mathbin{\Vert} e_2 \rightarrow e_1' \mathbin{\Vert} e_2}$
(OR-TRUE)	$\text{true} \mathbin{\Vert} e \rightarrow \text{true}$
(OR-FALSE)	$\text{false} \mathbin{\Vert} e \rightarrow e$

2.10. Imperative Konzepte

(BETA-V) $((\lambda \mathrm{id.e})\,\mathrm{v}, \sigma) \rightarrow (\mathrm{e}[\mathrm{v}/\mathrm{id}], \sigma)$

(APP-LEFT) $$\frac{(e_1, \sigma) \rightarrow (e_1', \sigma')}{(e_1 e_2, \sigma) \rightarrow (e_1' e_2, \sigma')}$$

(APP-RIGHT) $$\frac{(\mathrm{e}, \sigma) \rightarrow (\mathrm{e}', \sigma')}{(\mathrm{v\,e}, \sigma) \rightarrow (\mathrm{v\,e}', \sigma')}$$

(REF) $(\mathrm{ref\ v}, \sigma) \rightarrow (\mathrm{X}, \sigma[\mathrm{v}/\mathrm{X}])$ $\quad$ *mit* $X = \mathit{alloc}\ (\sigma)$

(ASSIGN) $(\mathrm{X} := \mathrm{v}, \sigma) \rightarrow ((), \sigma[\mathrm{v}/\mathrm{X}])$ $\quad$ *falls* $X \in \mathit{dom}\ (\sigma)$

(DEREF) $(!\mathrm{X}, \sigma) \rightarrow (\sigma(\mathrm{X}), \sigma)$ $\quad$ *falls* $X \in \mathit{dom}(\sigma)$

2.11. Objekte

(OBJECT) $$\frac{\mathrm{e}_{i+1} \rightarrow \mathrm{e}_{i+1}'}{\begin{array}{c}\textbf{object method } \mathrm{id}_1 = \mathrm{v}_1 \ldots \textbf{ method } \mathrm{id}_i = \mathrm{v}_i \textbf{ method } \mathrm{id}_{i+1} = \mathrm{e}_{i+1} \ldots \textbf{ end} \\ \rightarrow \textbf{object method } \mathrm{id}_1 = \mathrm{v}_1 \ldots \textbf{ method } \mathrm{id}_i = \mathrm{v}_i \textbf{ method } \mathrm{id}_{i+1} = \mathrm{e}_{i+1}' \ldots \textbf{ end}\end{array}}$$

(METHOD-EVAL) $$\frac{\mathrm{e} \rightarrow \mathrm{e}'}{\mathrm{e}\#\mathrm{id} \rightarrow \mathrm{e}'\#\mathrm{id}}$$

(METHOD-EXEC) $(\textbf{object method } \mathrm{id}_1 = \mathrm{v}_1 \ldots \textbf{ end})\#\mathrm{id}_i \rightarrow \mathrm{v}_i$ $\quad$ *für* $i = 1 \ldots n$

3. Big step Semantik

3.1. Werte

(VAL) $v \Downarrow v$

3.2. Operationen

(OP) $op\, n_1\, n_2 \Downarrow op^I(n_1, n_2)$

(INFIX) $$\frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{e_1\, op\, e_2 \Downarrow op^I(n_1, n_2)}$$

3.3. Beta Value

(BETA-V) $$\frac{e[v/id] \Downarrow v'}{(\lambda id.e)\, v \Downarrow v'}$$

3.4. Applikation

(APP) $$\frac{e_1 \Downarrow v_1 \qquad e_2 \Downarrow v_2 \qquad v_1\, v_2 \Downarrow v}{e_1\, e_2 \Downarrow v}$$

(APP*) $$\frac{e_1 \Downarrow v_1 \ldots e_n \Downarrow v_n \qquad v_1 \ldots v_n \Downarrow v}{e_1 \ldots e_n \Downarrow v}$$

3.5. Bedingungen

(COND-TRUE) $$\frac{e_0 \Downarrow \text{true} \qquad e_1 \Downarrow v}{\textbf{if}\ e_0\ \textbf{then}\ e_1\ \textbf{then}\ e_2 \Downarrow v}$$

(COND-FALSE) $$\frac{e_0 \Downarrow \text{false} \qquad e_2 \Downarrow v}{\textbf{if}\ e_0\ \textbf{then}\ e_1\ \textbf{else}\ e_2 \Downarrow v}$$

3.6. Let

(LET) $$\frac{e_1 \Downarrow v_1 \qquad e_2[v_1/id] \Downarrow v_2}{\textbf{let}\ id = e_1\ \textbf{in}\ e_2 \Downarrow v_2}$$

3.7. Rekursion

(UNFOLD) $$\frac{e[\textbf{rec}\ id.e/id] \Downarrow v}{\textbf{rec}\ id.e \Downarrow v}$$

3.8. Listen

(HD) $$\frac{e \Downarrow cons\ (v_1, v_2)}{hd\ (e) \Downarrow v_1}$$

(HD-EXN) $$\frac{e \Downarrow [\,]}{hd\ (e) \Downarrow empty\_list\_exn}$$

(TL) $$\frac{e \Downarrow cons\ (v_1, v_2)}{hd\ (e) \Downarrow v_2}$$

(TL-EXN) $$\frac{e \Downarrow [\,]}{tl\ (e) \Downarrow empty\_list\_exn}$$

(IS-EMPTY-TRUE) $$\frac{e \Downarrow [\,]}{is\_empty\ (e) \Downarrow true}$$

(IS-EMPTY-FALSE) $$\frac{e \Downarrow cons\ (v_1, v_2)}{is\_empty\ (e) \Downarrow false}$$

3.9. Paare

(FST) $fst\ (v_1, v_2) \Downarrow v_1$

(SND) $snd\ (v_1, v_2) \Downarrow v_2$

(PAIR) $$\frac{e_1 \Downarrow v_1 \quad e_2 \Downarrow v_2}{(e_1, e_2) \Downarrow (v_1, v_2)}$$

3.10. Exceptions

(EXN-i) $$\frac{e_1 \Downarrow v_1 \ \ \ldots\ \ e_{i-1} \Downarrow v_{i-1} \quad e_i \Downarrow exn}{e \Downarrow exn}$$

3.11. Boolsche Ausdrücke

3.11.1. And

(AND-TRUE) $$\frac{e_1 \Downarrow true \quad e_2 \Downarrow v}{e_1\ \&\&\ e_2 \Downarrow v}$$

(AND-FALSE) $$\frac{e_1 \Downarrow false}{e_1\ \&\&\ e_2 \Downarrow false}$$

3.11.2. Or

(OR-TRUE)

$$\frac{e_1 \Downarrow \text{true}}{e_1 \mathbin{||} e_2 \Downarrow \text{true}}$$

(OR-FALSE)

$$\frac{e_1 \Downarrow \text{false} \qquad e_2 \Downarrow v}{e_1 \mathbin{||} e_2 \Downarrow v}$$

3.11.3. Not

(NOT-TRUE)

$$\frac{e \Downarrow \text{true}}{\text{not } e \Downarrow \text{false}}$$

(NOT-FALSE)

$$\frac{e \Downarrow \text{false}}{\text{not } e \Downarrow \text{true}}$$

3.12. Imperative Konzepte

(VAL)

$$(v, \sigma) \Downarrow (v, \sigma)$$

(BETA-V)

$$\frac{(e[v/id], \sigma_0) \Downarrow (v', \sigma_1)}{((\lambda id.e)\, v, \sigma_0) \Downarrow (v', \sigma_1)}$$

(LET)

$$\frac{(e_1, \sigma_0) \Downarrow (v_1, \sigma_1) \qquad (e_2[v_1/id], \sigma_1) \Downarrow (v_2, \sigma_2)}{(\textbf{let}\, id = e_1 \,\textbf{in}\, e_2, \sigma_0) \Downarrow (v_2, \sigma_2)}$$

(APP)

$$\frac{(e_1, \sigma_0) \Downarrow (v_1, \sigma_1) \qquad (e_2, \sigma_1) \Downarrow (v_2, \sigma_2) \qquad (v_1 v_2, \sigma_2) \Downarrow (v, \sigma_3)}{(e_1 e_2, \sigma_0) \Downarrow (v, \sigma_3)}$$

(REF)

$$(\text{ref } v, \sigma) \Downarrow (X, \sigma[v/X]) \qquad \textit{mit } X = \textit{alloc}\,(\sigma)$$

(ASSIGN)

$$(X := v, \sigma) \Downarrow ((), \sigma[v/X]) \qquad \textit{falls } X \in \textit{dom}\,(\sigma)$$

(DEREF)

$$(!X, \sigma) \Downarrow (\sigma(X), \sigma) \qquad \textit{falls } X \in \textit{dom}\,(\sigma)$$

(SEQ)

$$\frac{(e_1, \sigma_0) \Downarrow (v_1, \sigma_1) \qquad (e_2, \sigma_1) \Downarrow (v_2, \sigma_2)}{(e_1 ; e_2, \sigma_0) \Downarrow (v_2, \sigma_2)}$$

(COND-1-TRUE)

$$\frac{(e_0, \sigma_0) \Downarrow (\text{true}, \sigma_1) \qquad (e_1, \sigma_1) \Downarrow (v, \sigma_2)}{(\textbf{if}\, e_0 \,\textbf{then}\, e_1, \sigma_0) \Downarrow (v, \sigma_2)}$$

(COND-1-FALSE)

$$\frac{(e_0, \sigma_0) \Downarrow (\text{false}, \sigma_1)}{(\textbf{if}\, e_0 \,\textbf{then}\, e_1, \sigma_0) \Downarrow ((), \sigma_1)}$$

(WHILE)

$$\frac{(\textbf{if}\, e_0 \,\textbf{then}\, e_1 ; \textbf{while}\, e_0 \,\textbf{do}\, e_1, \sigma_0) \Downarrow (v, \sigma_1)}{(\textbf{while}\, e_0 \,\textbf{do}\, e_1, \sigma_0) \Downarrow (v, \sigma_1)}$$

3.13. Objekte

(OBJECT)

$$\frac{e_1 \Downarrow v_1 \;\ldots\; e_n \Downarrow v_n}{\textbf{object method } \mathrm{id}_1 = e_1 \ldots \textbf{ method } \mathrm{id}_n = e_n \textbf{ end} \Downarrow \textbf{object method } \mathrm{id}_1 = v_1 \ldots \textbf{ method } \mathrm{id}_n = v_n \textbf{ end}}$$

(METHOD-EXEC)

$$\frac{e \Downarrow \textbf{object method } \mathrm{id}_1 = v_1 \ldots \mathrm{end}}{e\#\,\mathrm{id}_i \Downarrow v_i} \qquad \textit{für } i = 1 \ldots n$$

4. Call by name

4.1. Small step

	Call by value	Call by name

4.1.1. Beta

(BETA-V)	$(\lambda\, id.e)\, v \rightarrow e[v/id]$	*nicht vorhanden*
(BETA-E)	*nicht vorhanden*	$(\lambda\, id.e_1)\, e_2 \rightarrow e_1[e_2/id]$

4.1.2. Let

(LET-EVAL)	$\dfrac{e_1 \rightarrow e_1'}{\textbf{let}\ id = e_1\ \textbf{in}\ e_2 \rightarrow \textbf{let}\ id = e_1'\ \textbf{in}\ e_2}$	*nicht vorhanden*
(LET-EXEC)	$\textbf{let}\ id = v\ \textbf{in}\ e \rightarrow e[v/id]$	$\textbf{let}\ id = e_1\ \textbf{in}\ e_2 \rightarrow e_2[e_1/id]$

4.1.3. Applikation

(APP-RIGHT)	$\dfrac{e \rightarrow e'}{ve \rightarrow ve'}$	$\dfrac{e \rightarrow e'}{ve \rightarrow ve'}$ *falls v nicht von der Form $\lambda id.e_0$*

4.2. Big step

	Call by value	Call by name

4.2.1. Beta

(BETA-V)	$\dfrac{e[v/id] \Downarrow v'}{(\lambda id.e)\, v \Downarrow v'}$	*nicht vorhanden*
(BETA-E)	*nicht vorhanden*	$\dfrac{e_1[e_2/id] \Downarrow v}{(\lambda\, id.e_1) e_2 \Downarrow v}$

4.2.2. Let

(LET)	$\dfrac{e_1 \Downarrow v_1 \qquad e_2[v_1/id] \Downarrow v_2}{\textbf{let}\ id = e_1\ \textbf{in}\ e_2 \Downarrow v_2}$	$\dfrac{e_2[e_1/id] \Downarrow v}{\textbf{let}\ id = e_1\ \textbf{in}\ e_2 \Downarrow v}$

4.2.3. Applikation

(APP)	$\dfrac{e_1 \Downarrow v_1 \quad e_2 \Downarrow v_2 \quad v_1\, v_2 \Downarrow v}{e_1\, e_2 \Downarrow v}$	*nicht vorhanden*
(APP-LEFT)	*nicht vorhanden*	$\dfrac{e_1 \Downarrow v_1 \quad v_1\, e_2 \Downarrow v}{e_1\, e_2 \Downarrow v}$
(APP-RIGHT)	*nicht vorhanden*	$\dfrac{e_2 \Downarrow v_2 \quad v_1\, v_2 \Downarrow v}{v_1\, e_2 \Downarrow v}$ *falls v_1 nicht von der Form $\lambda id.e$*

5. Typsystem

5.1. Basistypen

(UNIT) $() :: \textbf{unit}$

(BOOL) $b :: \textbf{bool}$

(INT) $n :: \textbf{int}$

5.2. Operationen

(AOP) $op :: \textbf{int} \to \textbf{int} \to \textbf{int}$

(ROP) $op :: \textbf{int} \to \textbf{int} \to \textbf{bool}$

5.3. Konstanten

(CONST) $$\frac{c :: \tau}{\Gamma \triangleright c :: \tau}$$

5.4. Namen

(ID) $\Gamma \triangleright id :: \tau \quad \textit{falls } id \in dom(\Gamma) \textit{ und } \Gamma(id) = \tau$

(APP) $$\frac{\Gamma \triangleright e_1 :: \tau \to \tau' \quad \Gamma \triangleright e_2 :: \tau}{\Gamma \triangleright e_1\, e_2 :: \tau'}$$

5.5. Bedingungen

(COND) $$\frac{\Gamma \triangleright e_0 :: \textbf{bool} \quad \Gamma \triangleright e_1 :: \tau \quad \Gamma \triangleright e_2 :: \tau}{\Gamma \triangleright \textbf{if } e_0 \textbf{ then } e_1 \textbf{ else } e_2 :: \tau}$$

5.6. Let

(LET) $$\frac{\Gamma \triangleright e_1 :: \tau \quad \Gamma[\tau/id] \triangleright e_2 :: \tau'}{\Gamma \triangleright \textbf{let } id = e_1 \textbf{ in } e_2 :: \tau'}$$

5.7. Abstraktion

(ABSTR) $$\frac{\Gamma[\tau/id] \triangleright e :: \tau'}{\Gamma \triangleright \lambda\, id : \tau . e :: \tau \to \tau'}$$

(ABSTR') $$\frac{\Gamma[\tau/id] \triangleright e :: \tau'}{\Gamma \triangleright \lambda\, id . e :: \tau \to \tau'}$$

5.8. Rekursion

(REC) $$\frac{\Gamma[\tau/\mathrm{id}] \triangleright \mathrm{e}::\tau}{\Gamma \triangleright \mathbf{rec}\,\mathrm{id}:\tau.\mathrm{e}::\tau}$$

(REC') $$\frac{\Gamma[\tau/\mathrm{id}] \triangleright \mathrm{e}::\tau}{\Gamma \triangleright \mathbf{rec}\,\mathrm{id}.\mathrm{e}::\tau}$$

5.9. Listen

(EMPTY) $[]::\tau\ \mathbf{list}$

(CONS) $\mathrm{cons}::\tau * \tau\ \mathbf{list} \rightarrow \tau\ \mathbf{list}$

(HD) $\mathrm{hd}::\tau\ \mathbf{list} \rightarrow \tau$

(TL) $\mathrm{tl}::\tau\ \mathbf{list} \rightarrow \tau\ \mathbf{list}$

(IS-EMPTY) $\mathrm{is\_empty}::\tau\ \mathbf{list} \rightarrow \mathbf{bool}$

(LIST) $$\frac{\Gamma \triangleright \mathrm{e}_1::\tau \quad \ldots \quad \Gamma \triangleright \mathrm{e}_n::\tau}{\Gamma \triangleright [\mathrm{e}_1 \ldots \mathrm{e}_n]::\tau\ \mathbf{list}}$$

5.10. Paare

(FST) $\mathrm{fst}::\tau_1 * \tau_2 \rightarrow \tau_1$

(SND) $\mathrm{snd}::\tau_1 * \tau_2 \rightarrow \tau_2$

(PAIR) $$\frac{\Gamma \triangleright \mathrm{e}_1::\tau_1 \qquad \Gamma \triangleright \mathrm{e}_2::\tau_2}{\Gamma \triangleright (\mathrm{e}_1,\mathrm{e}_2)::\tau_1 * \tau_2}$$

5.11. Polymorphie

(P-LET) $$\frac{\Gamma \triangleright e_1::\tau \quad \Gamma[\mathrm{Closure}_{\Gamma}(\tau)/\mathrm{id}] \triangleright e_2::\tau'}{\Gamma \triangleright \mathbf{let}\ \mathrm{id}=e_1\ \mathbf{in}\ e_2::\tau'}$$

(P-ID) $$\Gamma \triangleright \mathrm{id}::\tau[\tau_1/\alpha_1 \ldots \tau_n/\alpha_n]$$
$$\textit{falls } id \in dom(\Gamma) \textit{ und } \Gamma(id) = \forall \alpha_1 \ldots \alpha_n . \tau$$

(P-CONST) $$\frac{c::\pi}{\Gamma \triangleright c::\tau} \quad \textit{falls } \tau \textit{ Instanz von } \pi$$

(P-EMPTY) $[]::\forall \alpha . \alpha\ \mathbf{list}$

(P-CONS) $\mathrm{cons}::\tau * \tau\ \mathrm{list} \rightarrow \tau\ \mathrm{list}$

(P-HD) $\mathrm{hd}::\forall \alpha . \alpha\ \mathbf{list} \rightarrow \alpha$

(P-TL) $\mathrm{tl}::\forall \alpha . \alpha\ \mathbf{list} \rightarrow \alpha\ \mathbf{list}$

(P-IS-EMPTY) $\mathrm{is\_empty}::\forall \alpha . \alpha\ \mathbf{list} \rightarrow \alpha\ \mathbf{list}$

(P-FST) $\mathrm{fst}::\forall \alpha_1, \alpha_2 . \alpha_1 * \alpha_2 \rightarrow \alpha_1$

(P-SND) $\mathrm{snd}::\forall \alpha_1, \alpha_2 . \alpha_1 * \alpha_2 \rightarrow \alpha_2$

5.12. Imperative Konzepte

(DEREF) $! :: \tau\ \mathrm{ref} \rightarrow \tau$

(ASSIGN) $:= :: \tau\ \mathrm{ref} \rightarrow \tau \rightarrow \mathrm{unit}$

(REF) $\mathrm{ref} :: \tau \rightarrow \tau\ \mathrm{ref}$

(SEQ) $$\frac{\Gamma \triangleright e_1::\tau_1 \quad \Gamma \triangleright e_2::\tau_2}{\Gamma \triangleright e_1 ; e_2::\tau_2}$$

(COND-1) $$\frac{\Gamma \triangleright e_0::\mathrm{bool} \quad \Gamma \triangleright e_1::\mathrm{unit}}{\Gamma \triangleright \mathbf{if}\ e_0\ \mathbf{then}\ e_1::\mathrm{unit}}$$

(WHILE) $$\frac{\Gamma \triangleright e_0::\mathrm{bool} \quad \Gamma \triangleright e_1::\tau}{\Gamma \triangleright \mathbf{while}\ e_0\ \mathbf{do}\ e_1::\mathrm{unit}}$$

(LOC) $\Sigma, \Gamma \triangleright X::\tau\ \mathrm{ref} \quad \textit{falls } X \in dom(\Sigma) \textit{ und } \Sigma(X) = \tau\ \textit{ref}$

5.13. Referenzen und Polymorphie

(P-LET-V) $$\frac{\Gamma \triangleright v::\tau \quad \Gamma[\mathrm{Closure}_{\Gamma}(\tau)/\mathrm{id}] \triangleright e::\tau'}{\Gamma \triangleright \mathbf{let}\ \mathrm{id}=v\ \mathbf{in}\ e::\tau'}$$

(LET-E) $$\frac{\Gamma \triangleright e_1::\tau \quad \Gamma[\tau/\mathrm{id}] \triangleright e_2::\tau'}{\Gamma \triangleright \mathbf{let}\ \mathrm{id}=e_1\ \mathbf{in}\ e_2::\tau'} \textit{falls } e \notin Val$$

5.14. Objekte

(OBJECT) $$\frac{\Gamma \triangleright e_1 :: \tau_1 \;\; \ldots \;\; \Gamma \triangleright e_n :: \tau_n}{\Gamma \triangleright \textbf{object method}\ id_1 = e_1 \ldots \textbf{method}\ id_n = e_n\ \textbf{end} :: < id_1 : \tau_1 ; \ldots ; id_n : \tau_n >}$$

(METHOD) $$\frac{\Gamma \triangleright e :: < id_1 : \tau_1 ; \ldots ; id_n : \tau_n >}{\Gamma \triangleright e \# id_i :: \tau_i} \quad \textit{für } i = 1 \ldots n$$

5.14.1. Subtyping

(SUBSUME) $$\frac{\tau_1 <: \tau_2 \quad \Gamma \triangleright e :: \tau_1}{\Gamma \triangleright e :: \tau_2}$$

(S-OBJ-WIDTH) $< id_1 : \tau_1 ; \ldots ; id_{n+k} : \tau_{n+k} > \;<:\; < id_1 : \tau_1 ; \ldots ; id_n : \tau_n >$

(S-OBJ-PERMUTE) $< id_1 : \tau_1 ; \ldots ; id_n : \tau_n > \;<:\; < id_{i_1} : \tau_{i_1} ; \ldots ; id_{i_n} : \tau_{i_n} > \quad \textit{falls } \{ i_1 \ldots i_n \} = \{ 1 \ldots n \}$

(S-OBJ-DEPTH) $$\frac{\tau_1 <: \tau_1' \;\; \ldots \;\; \tau_n <: \tau_n'}{< id_1 : \tau_1 ; \ldots ; id_n : \tau_n > \;<:\; < id_1 : \tau_1' ; \ldots ; id_n : \tau_n' >}$$

(S-REFL) $\tau <: \tau$

(S-TRANS) $$\frac{\tau_1 <: \tau_2 \quad \tau_2 <: \tau_3}{\tau_1 <: \tau_3}$$

(S-PRODUCT) $$\frac{\tau_1 <: \tau_1' \quad \tau_2 <: \tau_2'}{\tau_1 * \tau_2 <: \tau_1' * \tau_2'}$$

(S-LIST) $$\frac{\tau <: \tau'}{\tau\ \textbf{list} <: \tau'\ \textbf{list}}$$

(S-ARROW) $$\frac{\tau_1' <: \tau_1 \quad \tau_2 <: \tau_2'}{\tau_1 \to \tau_2 <: \tau_1' \to \tau_2'}$$

(S-ARROW-LEFT) $$\frac{\tau_1' <: \tau_1}{\tau_1 \to \tau_2 <: \tau_1' \to \tau_2}$$

(S-ARROW-RIGHT) $$\frac{\tau_2 <: \tau_2'}{\tau_1 \to \tau_2 <: \tau_1 \to \tau_2'}$$

(S-REF) $$\frac{\tau <: \tau' \quad \tau' <: \tau}{\tau\ \mathrm{ref} <: \tau'\ \mathrm{ref}}$$

6. Typinferenzalgorithmus

6.1. Teqns

6.1.1. Konstanten

(CONST) $\text{teqns}\ (\Gamma, \text{c}, \alpha) = \{\alpha = \tau\} \quad \textit{falls}\ c :: \tau$

6.1.2. Namen

(ID) $$\text{teqns}\ (\Gamma, \text{id}, \alpha) = \begin{cases} \{\alpha = \Gamma(\text{id})\} & \textit{falls}\ id \in dom(\Gamma) \\ \text{'unbekannter Name'} & \textit{sonst} \end{cases}$$

6.1.3. Applikation

(APP) $$\text{teqns}\ (\Gamma, \text{e}_1\text{e}_2, \alpha) = \\ \text{teqns}\ (\Gamma, \text{e}_1, \alpha_1) \cup \text{teqns}\ (\Gamma, \text{e}_2, \alpha_2) \cup \{\alpha_1 = \alpha_2 \rightarrow \alpha\}$$

6.1.4. Bedingungen

(COND) $$\text{teqns}\ (\Gamma, \textbf{if}\ \text{e}_0\ \textbf{then}\ \text{e}_1\ \textbf{else}\ \text{e}_{2,}\ \alpha) = \\ \text{teqns}\ (\Gamma, \text{e}_{0,}\ \alpha_0) \cup \text{teqns}\ (\Gamma, \text{e}_{1,}\ \alpha_1) \cup \\ \text{teqns}\ (\Gamma, \text{e}_{2,}\ \alpha_2) \cup \{\alpha_0 = \textbf{bool}, \alpha = \alpha_1, \alpha = \alpha_2\}$$

6.1.5. Let

(LET) $$\text{teqns}\ (\Gamma, \textbf{let}\ \text{id} = \text{e}_1\ \textbf{in}\ \text{e}_2, \alpha) = \\ \text{teqns}\ (\Gamma, \text{e}_1, \alpha_1) \cup \text{teqns}\ (\Gamma[\alpha_1/\text{id}], \text{e}_2, \alpha_2) \\ \cup \{\alpha = \alpha_2\}$$

6.1.6. Abstraktion

(ABSTR) $$\text{teqns}\ (\Gamma, \lambda\,\text{id.e}, \alpha) = \\ \text{teqns}\ (\Gamma[\alpha_1/\text{id}], \text{e}, \alpha_2) \cup \{\alpha = \alpha_1 \rightarrow \alpha_2\}$$

6.1.7. Rekursion

(REC) $$\text{teqns}\ (\Gamma, \textbf{rec}\ \text{id.e}, \alpha) = \\ \text{teqns}\ (\Gamma[\alpha_1/\text{id}], \text{e}, \alpha_2) \cup \{\alpha = \alpha_{1,}\ \alpha = \alpha_2\}$$

6.1.8. Paare

(PAIR) $$\text{teqns}\ (\Gamma, (\text{e}_1, \text{e}_2), \alpha) = \\ \text{teqns}\ (\Gamma, \text{e}_{1,}\ \alpha_1) \cup \text{teqns}\ (\Gamma, \text{e}_{2,}\ \alpha_2) \cup \{\alpha = \alpha_1 * \alpha_2\}$$

6.1.9. Listen

(LIST) $\text{teqns}\ (\Gamma, \text{cons}, \alpha) = \{\alpha = \alpha_1 * \alpha_1\ \text{list} \rightarrow \alpha_1\ \text{list}\}$

(EMPTY-LIST) $\text{teqns}\ (\Gamma, [], \alpha) = \{\alpha = \alpha_1\ \text{list}\}$

6.1.10. Imperative Konzepte

(DEREF) $\text{teqns}\ (\Gamma, !, \alpha) = \{\alpha = \alpha_1\,\text{ref} \to \alpha_1\}$

(ASSIGN) $\text{teqns}\ (\Gamma, :=, \alpha) = \{\alpha = \alpha_1\,\text{ref} \to \alpha_1 \to \text{unit}\}$

(REF) $\text{teqns}\ (\Gamma, \text{ref}, \alpha) = \{\alpha = \alpha_1 \to \alpha_1\,\text{ref}\}$

6.2. Unify

$\text{unify}\ (\emptyset) = [\,]$

$\text{unify}\ (\{\tau = \tau\} \cup \text{E}) = \text{unify}\ (\text{E})$

$$\text{unify}\ (\{\alpha = \tau\} \cup \text{E}) = \text{unify}\ (\{\tau = \alpha\} \cup \text{E})$$
$$= \begin{cases} \text{s}_1\text{s}_2 \text{ mit } \text{s}_1 = [\tau/\alpha] \text{ und } \text{s}_2 = \text{unify}\ (\text{E}\,\text{s}_1) & \textit{falls}\ \alpha \notin \textit{tvar}\ (\tau) \\ \text{'nicht unifizierbar'} & \textit{falls}\ \alpha \in \textit{tvar}\ (\tau)\ \textit{und}\ \ \alpha \neq \tau \end{cases}$$

$\text{unify}\ (\{\tau_1 \to \tau_2 = \tau_1' \to \tau_2'\} \cup \text{E}) = \text{unify}\ (\{\tau_1 = \tau_1',\ \tau_2 \to \tau_2'\} \cup \text{E})$

$\text{unify}\ (\{\tau_1 = \tau_2\} \cup \text{E}) =$ *'nicht unifizierbar' in allen anderen Fällen*

6.2.1. Imperative Konzepte

$\text{unify}\ (\{\tau_1\,\text{ref} = \tau_2\,\text{ref}\} \cup \text{E}) = \text{unify}\ (\{\tau_1 = \tau_2\} \cup \text{E})$

6.3. Solution

6.3.1. Konstanten

$\text{solution}\ (\Gamma, \text{c}, \tau) = \text{unify}\ (\{\tau = \tau'\})$ *falls* $c :: \pi$ *und* τ' *neue Instanz von* π

6.3.2. Namen

$\text{solution}\ (\Gamma, \text{id}, \tau) = \text{unify}\ (\{\tau = \tau'\})$ *falls* $id \in dom(\Gamma)$ *und* τ' *neue Instanz von* $\Gamma\ (id)$

6.3.3. Applikation

$\text{solution}\ (\Gamma, \text{e}_1\,\text{e}_2, \tau) = \text{s}_1\text{s}_2$
$\text{s}_1 = \text{solution}\ (\Gamma, \text{e}_1, \alpha \to \tau)$ *mit neuer Typvariablen* α
$\text{s}_2 = \text{solution}\ (\Gamma\,\text{s}_1, \text{e}_2, \alpha\,\text{s}_1)$

6.3.4. Bedingungen

$\text{solution}\ (\Gamma, \textbf{if}\ \text{e}_0\ \textbf{then}\ \text{e}_1\ \textbf{else}\ \text{e}_2, \tau) = \text{s}_0\text{s}_1\text{s}_2$
$\text{s}_0 = \text{solution}\ (\Gamma, \text{e}_0, \textbf{bool})$
$\text{s}_1 = \text{solution}\ (\Gamma\,\text{s}_0, \text{e}_1, \tau\,\text{s}_0)$
$\text{s}_2 = \text{solution}\ (\Gamma\,\text{s}_0\,\text{s}_1, \text{e}_2, \tau\,\text{s}_0\,\text{s}_1)$

6.3.5. Paare

$\text{solution}\ (\Gamma, (e_1, e_2), \tau) = s_1 s_2 s_3$
$s_1 = \text{solution}\ (\Gamma, e_1, \alpha_1)$ *mit neuer Typvariablen* α_1
$s_2 = \text{solution}\ (\Gamma s_1, e_2, \alpha_2 s_1)$ *mit neuer Typvariablen* α_2
$s_3 = \text{unify}\ (\tau s_1 s_2 = \alpha_1 s_1 s_2 * \alpha_2 s_1 s_2)$

6.3.6. Let

$\text{solution}\ (\Gamma, \textbf{let}\ \text{id} = e_1\ \textbf{in}\ e_2, \tau) = s_1 s_2$
$s_1 = \text{solution}\ (\Gamma, e_1, \alpha)$ *mit neuer Typvariablen* α
$s_2 = \text{solution}\ (\Gamma s_1 [\text{Closure}_{\Gamma s_1}(\alpha s_1)/\text{id}], e_2, \tau s_1)$

6.3.7. Abstraktion

$\text{solution}\ (\Gamma, \lambda\, \text{id}.e, \tau) = s_1 s_2$
$s_1 = \text{solution}\ (\Gamma[\alpha_1/\text{id}], e, \alpha_2)$ *mit neuen Typvariablen* α_1, α_2
$s_2 = \text{unify}\ (\{\tau s_1 = \alpha_1 s_1 \rightarrow \alpha_2 s_1\})$

6.3.8. Rekursion

$\text{solution}\ (\Gamma, \textbf{rec}\ \text{id}.e, \tau) = \text{solution}\ (\Gamma[\tau/\text{id}], e, \tau)$