Übung zur Vorlesung

Automatisierte Logik und Programmierung I

Prof. Chr. Kreitz

Universität Potsdam, Theoretische Informatik — Wintersemester 2013/14

Blatt 4 — Abgabetermin: 29.11.2013

Das vierte Übungsblatt soll dazu dienen, sich mit Typisierung, Typ-Inferenzalgorithmen und der Argumentation in Normalisierungsbeweisen auseinanderzusetzen.
Wir werden mögliche Lösungen zu Beginn der Veranstaltung am 29.11.2013 besprechen.

Aufgabe 4.1 (Typisierung)

Geben Sie, wo möglich, eine Typisierung für die folgenden Terme an.

4.1–a $\lambda \mathtt{t}.\lambda \mathtt{y}.\ \mathtt{t\,y\,y}$

4.1–b $(\lambda \mathtt{x}.\lambda \mathtt{y}.\ \mathtt{x\,y})\ (\lambda \mathtt{z}.\ \mathtt{z})$

4.1–c $\lambda \mathtt{f}.\ (\lambda \mathtt{x}.\ \mathtt{f}\ (\mathtt{x\,x}))\ (\lambda \mathtt{x}.\ \mathtt{f}\ (\mathtt{x\,x}))$

4.1–d $\lambda \mathtt{y}.\lambda \mathtt{g}.\ (\lambda \mathtt{x}.\ \mathtt{x}^3\ \mathtt{y})\ (\lambda \mathtt{z}.\ \mathtt{g\,z\,z})$

Welches Ergebnis würde eine Anwendung des Hindley–Milner–Algorithmus auf diese Terme liefern?

4.1-e Zeigen Sie durch Induktion, daß die Church–Numerals ($\overline{n} \equiv \lambda \mathtt{f}.\lambda \mathtt{x}.\ \mathtt{f}^n\ \mathtt{x}$) für alle $n \in \mathbb{N}$ mit $(\mathtt{X}\to\mathtt{X})\to\mathtt{X}\to\mathtt{X}$ typisiert werden können.

Aufgabe 4.2 (Lehren aus der Leere ziehen)

Der Beweis der starken Normalisierbarkeit typisierbarer λ-Terme setzt implizit voraus, daß Datentypen nicht leer sind.

Nehmen Sie an, es gäbe einen leeren Datentyp $\{\}$ und das explizite Urteil $x \in \{\}$ *gilt niemals*. Zeigen Sie anhand eines Beispiels, daß es nun typisierbare Terme gibt, die nicht stark normalisierbar sind.

Untersuchen Sie anhand dieses Beispiels, an welcher Stelle der Beweis der starken Normalisierbarkeit fehlschlägt, wenn Typisierung mit leeren Datentypen erlaubt ist.

Aufgabe 4.3 (Hindley–Milner Algorithmus in der Praxis)

In vielen funktionalen Programmiersprachen wie ML oder Haskell wird eine erweiterte Version des Hindley–Milner Typechecking Algorithmus eingesetzt um den Datentyp eines gegebenen Ausdrucks der Sprache zu bestimmen.

Wie müsste der Hindley–Milner Algorithmus erweitert werden, wenn neben dem einfachen Funktionenraum auch die folgenden Datenstrukturen zum Typsystem der Sprache gehören.

- Den Typ $\mathbb{B}$ der booleschen Werte zusammen mit den Elementen T und F und der Analyseoperation $\mathsf{if}\ b\ \mathsf{then}\ s\ \mathsf{else}\ t$.
- Das Produkt $S\times T$ zweier Datentypen S und T zusammen mit dem Element $\langle s, t\rangle$ (Paarbildung) und der Analyseoperation $\mathsf{let}\ \langle x, y\rangle = p\ \mathsf{in}\ e$.
- Den Typ $\mathbb{N}$ der natürlichen Zahlen mit den Element $\mathbf{0}$, der Nachfolgeroperation $\mathsf{s}(i)$, den arithmetischen Ausdrücken i+j, i-j, i*j, i/j, $i\ \mathsf{mod}\ j$, und einem induktiven Analyseoperator $\mathsf{PR}[base; h](i)$[1].
- Den Typ $T\ \mathsf{list}$ der Listen über dem Datentyp T zusammen mit den Element $[]$, der Operation t::l (t wird an den Anfang der Liste l gehängt) und einem induktiven Analyseoperator $\mathsf{list\_ind}[base; h](l)$[2].

---

[1] Oft geschrieben als `f(i), where f(x) = if x=0 then base else h(x,f(x-1))`

[2] Oft geschrieben als `f(l), where f(x) =if x=[] then base else let x=hd::tl in h(hd,tl,f(tl))`

Lösung 4.1 *Ziel dieser Aufgabe ist es, sich ein wenig auf das Typenkonzept und die zugehörigen Berechnungen einzustimmen. Es werden dazu einige Beispiele von niederer Schwierigkeit dargeboten:*

4.1–a λt.λy. t y y: Die Analyse von Hand ergibt $(S \to S \to T) \to S \to T$

Env	*Aktueller Term*	σ	UNIFY	*Typ*
	λt.λy.t y y			
t:X_0	λy.t y y			
t:X_0, y:X_1	t y y			
t:X_0, y:X_1	t y			
t:X_0, y:X_1	t			X_0
t:X_0, y:X_1	y			X_1
t:X_0, y:X_1	t y		X_0=$X_1 \to X_2$	X_2
t:X_0, y:X_1	y	[$X_1 \to X_2$/X_0]		X_1
t:X_0, y:X_1	t y y	[$X_1 \to X_2$/X_0]	X_2=$X_1 \to X_3$	X_3
t:X_0, y:X_1	λy.t y y	[$X_1 \to X_3$, $X_1 \to X_2$/X_2, X_0]		$X_1 \to X_3$
t:X_0, y:X_1	λt.λy.t y y	[$X_1 \to X_3$, $X_1 \to X_2$/X_2, X_0]		$(X_1 \to X_1 \to X_3) \to X_1 \to X_3$

4.1–b (λx.λy. x y) (λz. z): Die Analyse von Hand ergibt $S \to S$

Env	*Aktueller Term*	σ	UNIFY	*Typ*
	(λx.λy. x y) (λz. z)			
	λx.λy. x y			
x:X_0	λy. x y			
x:X_0, y:X_1	x y			
x:X_0, y:X_1	x			X_0
x:X_0, y:X_1	y			X_1
x:X_0, y:X_1	x y		X_0 = $X_1 \to X_2$	X_2
x:X_0, y:X_1	λy. x y	[$X_1 \to X_2$/X_0]		$X_1 \to X_2$
x:X_0, y:X_1	λx.λy. x y	[$X_1 \to X_2$/X_0]		$(X_1 \to X_2) \to X_1 \to X_2$
x:X_0, y:X_1	λz. z	[$X_1 \to X_2$/X_0]		
x:X_0, y:X_1, z:X_3	z	[$X_1 \to X_2$/X_0]		X_3
x:X_0, y:X_1, z:X_3	λz. z	[$X_1 \to X_2$/X_0]		$X_3 \to X_3$
x:X_0, y:X_1, z:X_3	(λx.λy. x y) (λz. z)	[$X_1 \to X_2$/X_0]	$(X_1 \to X_2) \to X_1 \to X_2 = (X_3 \to X_3) \to X_4$	X_4
x:X_0, y:X_1, z:X_3	(λx.λy. x y) (λz. z)	$\begin{bmatrix} X_3 \to X_3/X_4 \\ X_3/X_1 \\ X_3/X_2 \\ X_3 \to X_3/X_0 \end{bmatrix}$		$X_3 \to X_3$

4.1–c *Ziel dieser Teilaufgabe ist es, sich einmal klarzumachen, warum ein "Metakonstrukt" wie der Y–Kombinator nicht typisierbar sein kann.*

λf. (λx. f (x x)) (λx. f (x x)): *Wegen* x x *darf der Term nicht typisierbar sein*

Env	*Aktueller Term*	σ	UNIFY	*Typ*
	λf. (λx. f (x x)) (λx. f (x x))			
f:X_0	(λx. f (x x)) (λx. f (x x))			
f:X_0	λx. f (x x)			
f:X_0, x:X_1	f (x x)			
f:X_0, x:X_1	f			X_0
f:X_0, x:X_1	x x			
f:X_0, x:X_1	x			X_1
f:X_0, x:X_1	x			X_1
f:X_0, x:X_1	x x		X_1=$X_1 \to X_2$	X_2
f:X_0, x:X_1	f (x x)	[??/X_1]	X_0=$X_2 \to X_3$	X_3

Hier müßte ein Typ durch einen Funktionstyp ersetzt werden, in welchem er selbst vorkommt. Dies würde zu einer unendlichen Ersetzung führen, die somit nicht berechenbar ist Die Unifikation schlägt deshalb fehl. Tatsächlich wird der Kombinator typisierbar, sobald er auf einen typisierbaren Term angewendet wird, für den es einen Fixpunkt gibt.

4.1–d $\lambda y.\lambda g.\ (\lambda x.\ x^3\,y)\ (\lambda z.\ g\,z\,z)$: Die Analyse von Hand ergibt $y\in Y$, $g\in(S\to S\to T)$, $\lambda z.\,g\,z\,z\in S\to T$, $\lambda x.\,x^3\,y\in Y\to Y\to Y$, damit $S=T=Y$ und Gesamttyp $Y\to(Y\to Y\to Y)\to Y$

Env	*Aktueller Term*	σ	UNIFY	*Typ*
	$\lambda y.\lambda g.(\lambda x.\,x^3\,y)\ (\lambda z.\,g\,z\,z)$			
$y{:}X_0$	$\lambda g.(\lambda x.\,x^3\,y)\ (\lambda z.\,g\,z\,z)$			
$y{:}X_0,\ g{:}X_1$	$(\lambda x.\,x^3\,y)\ (\lambda z.\,g\,z\,z)$			
$y{:}X_0,\ g{:}X_1$	$\lambda x.\,x^3\,y$			
$y{:}X_0,\ g{:}X_1,\ x{:}X_2$	$x^3\,y$			
$y{:}X_0,\ g{:}X_1,\ x{:}X_2$	x			X_2
$y{:}X_0,\ g{:}X_1,\ x{:}X_2$	$x^2\,y$			
$y{:}X_0,\ g{:}X_1,\ x{:}X_2$	x			X_2
$y{:}X_0,\ g{:}X_1,\ x{:}X_2$	$x\,y$			
$y{:}X_0,\ g{:}X_1,\ x{:}X_2$	x			X_2
$y{:}X_0,\ g{:}X_1,\ x{:}X_2$	y			X_0
$y{:}X_0,\ g{:}X_1,\ x{:}X_2$	$x\,y$		$X_2=X_0\to X_3$	X_3
$y{:}X_0,\ g{:}X_1,\ x{:}X_2$	$x^2\,y$	$[X_0\to X_3/X_2]$	$X_0\to X_3=X_3\to X_4$	X_4
$y{:}X_0,\ g{:}X_1,\ x{:}X_2$	$x^3\,y$	$[X_0\to X_3/X_2,\ X_4/X_0,\ X_4/X_3]$	$X_4\to X_4=X_4\to X_5$	X_5
$y{:}X_0,\ g{:}X_1,\ x{:}X_2$	$\lambda x.\,x^3\,y$	$[X_0\to X_3/X_2,\ X_4/X_0,\ X_4/X_3,\ X_5/X_4]$		$(X_5\to X_5)\to X_5$
$y{:}X_0,\ g{:}X_1,\ x{:}X_2$	$\lambda z.\,g\,z\,z$	$[X_0\to X_3/X_2,\ X_4/X_0,\ X_4/X_3,\ X_5/X_4]$		
$y{:}X_0,\ g{:}X_1,\ x{:}X_2,\ z{:}X_6$	$g\,z\,z$	$[X_0\to X_3/X_2,\ X_4/X_0,\ X_4/X_3,\ X_5/X_4]$		
$y{:}X_0,\ g{:}X_1,\ x{:}X_2,\ z{:}X_6$	$g\,z$	$[X_0\to X_3/X_2,\ X_4/X_0,\ X_4/X_3,\ X_5/X_4]$		
$y{:}X_0,\ g{:}X_1,\ x{:}X_2,\ z{:}X_6$	g	$[X_0\to X_3/X_2,\ X_4/X_0,\ X_4/X_3,\ X_5/X_4]$		X_1
$y{:}X_0,\ g{:}X_1,\ x{:}X_2,\ z{:}X_6$	z	$[X_0\to X_3/X_2,\ X_4/X_0,\ X_4/X_3,\ X_5/X_4]$		X_6
$y{:}X_0,\ g{:}X_1,\ x{:}X_2,\ z{:}X_6$	$g\,z$	$[X_0\to X_3/X_2,\ X_4/X_0,\ X_4/X_3,\ X_5/X_4]$	$X_1=X_6\to X_7$	X_7
$y{:}X_0,\ g{:}X_1,\ x{:}X_2,\ z{:}X_6$	z	$[X_0\to X_3/X_2,\ X_4/X_0,\ X_4/X_3,\ X_5/X_4,\ X_6\to X_7/X_1]$		X_6
$y{:}X_0,\ g{:}X_1,\ x{:}X_2,\ z{:}X_6$	$g\,z\,z$	$[X_0\to X_3/X_2,\ X_4/X_0,\ X_4/X_3,\ X_5/X_4,\ X_6\to X_7/X_1]$	$X_7=X_6\to X_8$	X_8
$y{:}X_0,\ g{:}X_1,\ x{:}X_2,\ z{:}X_6$	$\lambda z.\,g\,z\,z$	$[X_0\to X_3/X_2,\ X_4/X_0,\ X_4/X_3,\ X_5/X_4,\ X_6\to X_7/X_1,\ X_6\to X_8/X_7]$		$X_6\to X_8$
$y{:}X_0,\ g{:}X_1,\ x{:}X_2,\ z{:}X_6$	$(\lambda x.\,x^3\,y)\ (\lambda z.\,g\,z\,z)$	$[X_0\to X_3/X_2,\ X_4/X_0,\ X_4/X_3,\ X_5/X_4,\ X_6\to X_7/X_1,\ X_6\to X_8/X_7]$	$(X_5\to X_5)\to X_5 = (X_6\to X_8)\to X_9$	X_9
$y{:}X_0,\ g{:}X_1,\ x{:}X_2,\ z{:}X_6$	$(\lambda x.\,x^3\,y)\ (\lambda z.\,g\,z\,z)$	$[X_0\to X_3/X_2,\ X_4/X_0,\ X_4/X_3,\ X_5/X_4,\ X_6\to X_7/X_1,\ X_6\to X_8/X_7,\ X_5/X_6,\ X_5/X_8,\ X_5/X_9]$		X_5

4.1–e **Typisierung von $\overline{0}$ mit** $(\mathtt{X}\to\mathtt{X})\to\mathtt{X}\to\mathtt{X}$:

Env	*Aktueller Term*	σ	UNIFY	*Typ*
	λf.λx. x			
f:X_0	λx. x			
f:X_0, x:X_1	x			X_1
f:X_0, x:X_1	λx. x			$X_1\to X_1$
f:X_0, x:X_1	λf.λx. x			$X_0\to X_1\to X_1$

Setze nun einfach X_0=$X_1\to X_1$ sowie X_1=X.

Typisierung von $\overline{1}$ mit $(\mathtt{X}\to\mathtt{X})\to\mathtt{X}\to\mathtt{X}$:

Wurde bereits in 4.1–b gezeigt (ersetze dort X_1 und X_2 jeweils durch X).

Typisierung von $\overline{n+1}$ mit $(\mathtt{X}\to\mathtt{X})\to\mathtt{X}\to\mathtt{X}$**, falls $\overline{n}$ mit** $(\mathtt{X}\to\mathtt{X})\to\mathtt{X}\to\mathtt{X}$ **typisierbar ist**:

Sei λf.λx. $\mathtt{f}^n$ x vom Typ $(\mathtt{X}\to\mathtt{X})\to\mathtt{X}\to\mathtt{X}$. Dann gelten folgende Typisierungen:

1. f ist vom Typ X→X
2. x ist vom Typ X
3. $\mathtt{f}^n$ x ist vom Typ X

Wir typisieren nun λf.λx. $\mathtt{f}^{n+1}$ x $\equiv$ λf.λx. f ($\mathtt{f}^n$ x) unter Beachtung dieser Bedingungen:

Env	*Aktueller Term*	σ	UNIFY	*Typ*
f:X→X, x:X, $\mathtt{f}^n$ x:X	λf.λx. f ($\mathtt{f}^n$ x)			
f:X→X, x:X, $\mathtt{f}^n$ x:X	λx. f ($\mathtt{f}^n$ x)			
f:X→X, x:X, $\mathtt{f}^n$ x:X	f ($\mathtt{f}^n$ x)			
f:X→X, x:X, $\mathtt{f}^n$ x:X	f			X→X
f:X→X, x:X, $\mathtt{f}^n$ x:X	($\mathtt{f}^n$ x)			X
f:X→X, x:X, $\mathtt{f}^n$ x:X	f ($\mathtt{f}^n$ x)		X→X=X→X_0	X_0
f:X→X, x:X, $\mathtt{f}^n$ x:X	λx. f ($\mathtt{f}^n$ x)	[X/X_0]		X→X
f:X→X, x:X, $\mathtt{f}^n$ x:X	λf.λx. f ($\mathtt{f}^n$ x)	[X/X_0]		(X→X)→X→X

Lösung 4.2 Betrachte den Term $t = \lambda z.(\lambda x.\,x\,x)(\lambda x.\,x\,x)$, der bekanntlich nicht terminiert.

Dann gilt $t\in\{\}\to T$ für jeden beliebigen Typ T, denn das Urteil der Typzugehörigkeit von $\lambda x\,.\,t\in\{\}\to T$ verlangt nur, daß $t\in T$ aus $x\in\{\}$ folgt. Da wir aber explizit fordern "$x\in\{\}$ *gilt niemals*" ist diese Folgerung gültig und damit $\lambda x\,.\,t\in\{\}\to T$.

Damit haben wir eine Typisierung für einen nicht-normalisierenden Term angegeben, was bedeutet, daß nicht alle typisierbaren Terme stark normalisierbar sind.

Im Beweis der starken Normalisierbarkeit typisierbarer λ-Terme müssen wir nun nach einer Stelle suchen, in der wir implizit die Existenz eines $x\in T$ vorausgesetzt haben und nun $T = \{\}$ verwenden. Man könnte hier den Schritt der Behauptung 1 anführen, in dem wir Variablen vom Typ T_1 als (Platzhalter für) berechenbare Objekte des Typs T_1 betrachten. $\{\}$ besitzt aber keine Objekte.

Der Beweis müsste an dieser Stelle präzisiert werden zu "x Variable für die $x\in T_1$ gilt". Dann wird klar, daß wir hierfür einen "Nachweis" benötigen. Etwas sauberer wäre es, anstelle einer Variablen vom Typ T_1 einen neutralen Term s vom Typ T_1 zu verwenden, der in Normalform ist, und explizit vorauszusetzen, daß es solche Terme in T_1 überhaupt gibt (da man sonst s gar nicht verwenden darf).

Lösung 4.3

- **Eingabe**: geschlossener Term t
  Ausgabe: prinzipielles Typschema von t oder Fehlermeldung
- **Start**: Setze globale Substitution σ := [] und rufe TYPE-OF([], t) auf
- **Algorithmus** TYPE-OF(Env, t):
  Falls $t = x \in \mathcal{V}$: suche Deklaration $x{:}T$ in Env Ausgabe: T
  Falls $t = f\ u$: Setze S_1:=TYPE-OF(Env, f), S_2:=TYPE-OF(Env, u)
  Wähle neues X_{i+1} und unifiziere $\sigma(S_1)$ mit $S_2 \rightarrow X_{i+1}$.
  Fehlermeldung, wenn Unifikation fehlschlägt.
  Sonst $\sigma := \sigma' \circ \sigma$, wobei σ' Ergebnis der Unifikation Ausgabe: $\sigma(X_{i+1})$
  Falls $t = \lambda x.u$: Wähle neues X_{i+1}
  $S_1 := \begin{cases} \texttt{TYPE-OF}(Env\cdot[x{:}X_{i+1}], u) & \text{falls } x \text{ nicht in } Env \\ \texttt{TYPE-OF}(Env\cdot[x'{:}X_{i+1}], u[x'/x]) & \text{sonst } (x' \in \mathcal{V} \text{ neu}) \end{cases}$ Ausgabe: $\sigma(X_{i+1}) \rightarrow S_1$
  Falls t = T oder t = F: Ausgabe: $\mathbb{B}$
  Falls t = if b then s else t:
  Setze S_1:=TYPE-OF(Env, b), S_2:=TYPE-OF(Env, s), S_3:=TYPE-OF(Env, t)
  Unifiziere $\sigma(S_1)$ mit $\mathbb{B}$. Fehlermeldung, wenn Unifikation fehlschlägt.
  Sonst $\sigma := \sigma' \circ \sigma$, wobei σ' Ergebnis der Unifikation
  Unifiziere $\sigma(S_2)$ mit $\sigma(S_3)$. Fehlermeldung, wenn Unifikation fehlschlägt.
  Sonst $\sigma := \sigma' \circ \sigma$, wobei σ' Ergebnis der Unifikation Ausgabe: $\sigma(S_3)$
  Falls $t = \langle s, t\rangle$: Setze S_1:=TYPE-OF(Env, s), S_2:=TYPE-OF(Env, t) Ausgabe: $S_1 \times S_2$
  Falls t = let $\langle x, y\rangle = p$ in e: Wähle neues X_{i+1} und X_{i+2}
  Setze S_1:=TYPE-OF(Env, p)
  Unifiziere $\sigma(S_1)$ mit $X_{i+1} \times X_{i+2}$. Fehlermeldung, wenn Unifikation fehlschlägt.
  Sonst $\sigma := \sigma' \circ \sigma$, wobei σ' Ergebnis der Unifikation
  $S_2 := \begin{cases} \texttt{TYPE-OF}(Env\cdot[x{:}X_{i+1};\ y{:}X_{i+2}], e) & \text{falls } x, y \text{ nicht in } Env \\ \texttt{TYPE-OF}(Env\cdot[x'{:}X_{i+1};\ y'{:}X_{i+2}]], e[x', y'/x, y]) & \text{sonst } (x', y' \in \mathcal{V} \text{ neu}) \end{cases}$
  Ausgabe: $\sigma(S_2)$
  Falls t = 0: Ausgabe: $\mathbb{N}$
  Falls t = s(i): Setze S_1:=TYPE-OF(Env, i)
  Unifiziere $\sigma(S_1)$ mit $\mathbb{N}$. Fehlermeldung, wenn Unifikation fehlschlägt.
  Sonst $\sigma := \sigma' \circ \sigma$, wobei σ' Ergebnis der Unifikation Ausgabe: $\mathbb{N}$
  Falls $t = i$+j oder $t = i$-j oder $t = i$*j oder $t = i/j$ oder $t = i$ mod j:
  Setze S_1:=TYPE-OF(Env, i), S_2:=TYPE-OF(Env, j)
  Unifiziere $\sigma(S_1)$ mit $\mathbb{N}$. Fehlermeldung, wenn Unifikation fehlschlägt.
  Sonst $\sigma := \sigma' \circ \sigma$, wobei σ' Ergebnis der Unifikation
  Unifiziere $\sigma(S_2)$ mit $\mathbb{N}$. Fehlermeldung, wenn Unifikation fehlschlägt.
  Sonst $\sigma := \sigma' \circ \sigma$, wobei σ' Ergebnis der Unifikation Ausgabe: $\mathbb{N}$
  Falls t = PR[$base$; h](i): Setze S_1:=TYPE-OF(Env, i), S_2:=TYPE-OF(Env, $base$), S_3:=TYPE-OF(Env, h)
  Unifiziere $\sigma(S_1)$ mit $\mathbb{N}$. Fehlermeldung, wenn Unifikation fehlschlägt.
  Sonst $\sigma := \sigma' \circ \sigma$, wobei σ' Ergebnis der Unifikation
  Unifiziere $\sigma(S_3)$ mit $\sigma(\mathbb{N} \rightarrow S_2 \rightarrow S_2)$. Fehlermeldung, wenn Unifikation fehlschlägt.
  Sonst $\sigma := \sigma' \circ \sigma$, wobei σ' Ergebnis der Unifikation Ausgabe: $\sigma(S_2)$
  Falls t = []: Wähle neues X_{i+1} Ausgabe: X_{i+1} list
  Falls $t = t{::}l$:
  Setze S_1:=TYPE-OF(Env, t), S_2:=TYPE-OF(Env, l)
  Unifiziere $\sigma(S_2)$ mit $\sigma(S_1)$ list. Fehlermeldung, wenn Unifikation fehlschlägt.
  Sonst $\sigma := \sigma' \circ \sigma$, wobei σ' Ergebnis der Unifikation Ausgabe: $\sigma(S_2)$
  Falls t = list_ind[$base$; h](l):
  Setze S_1:=TYPE-OF(Env, l), S_2:=TYPE-OF(Env, $base$), S_3:=TYPE-OF(Env, h)
  Wähle neues X_{i+1}
  Unifiziere $\sigma(S_1)$ mit X_{i+1} list. Fehlermeldung, wenn Unifikation fehlschlägt.
  Sonst $\sigma := \sigma' \circ \sigma$, wobei σ' Ergebnis der Unifikation
  Unifiziere $\sigma(S_3)$ mit $\sigma(X_{i+1} \rightarrow X_{i+1}$ list $\rightarrow S_2 \rightarrow S_2)$. Fehlermeldung, wenn Unifikation fehlschlägt.
  Sonst $\sigma := \sigma' \circ \sigma$, wobei σ' Ergebnis der Unifikation Ausgabe: $\sigma(S_2)$