

3. Mathematische Grundlagen

Links zu den Vorträgen von Peter Mirolid:

- Aussagenlogik: <http://rw4.cs.uni-sb.de/~joba/Info1/Material/mathe1.pdf>
- Prädikatenlogik: http://rw4.cs.uni-sb.de/~joba/Info1/Material/Math_Grundl_2.pdf
- Relationen: <http://rw4.cs.uni-sb.de/~joba/Info1/Material/Relationen.pdf>

Die reflexive, transitive Hülle

$$R_0 = \{R' \mid R' \subseteq A \times A, R' \text{ reflexiv, transitiv}, R \subseteq R'\}$$

$$\hat{R} = \bigcap_{R' \in R_0} R' \quad \text{„von oben“}$$

$$R^* = \{(a, b) \in A \times A \mid \text{es gibt } a_0, a_1, \dots, a_n \in A \text{ mit } a = a_0, b = a_n \text{ und } (a_i, a_{i+1}) \in R \text{ für } (0 \leq i < n)\} \\ \text{„von unten“}$$

Satz: \hat{R} ist reflexiv und transitiv

1. Beweis zu reflexiv:

Sei $a \in A$ beliebig, $(a, a) \in R'$ für alle $R' \in R_0$

Es folgt $(a, a) \in \bigcap_{R' \in R_0} R' = \hat{R}$

2. Beweis zu transitiv:

Seien a, b, c beliebig aus A mit $(a, b) \in \hat{R}$ und $(b, c) \in \hat{R}$

$$\left. \begin{array}{l} (a, b) \in \hat{R} \Rightarrow (a, b) \in R' \text{ für alle } R' \in R_0 \\ (b, c) \in \hat{R} \Rightarrow (b, c) \in R' \text{ für alle } R' \in R_0 \end{array} \right\} (a, c) \in R' \text{ für alle } R' \in R_0$$

Daraus folgt $(a, c) \in \bigcap_{R' \in R_0} R' = \hat{R}$

Satz: $R^* = \hat{R}$

Wir zeigen, $R^* \subseteq \hat{R}$ und $\hat{R} \subseteq R^*$

2. Beweis zu $\hat{R} \subseteq R^*$:

Hilfsbehauptungen:

- 2.1. R^* ist reflexiv
- 2.2. R^* ist transitiv
- 2.3. $R \subseteq R^*$

Aus diesen folgt:

$$R^* \in R_0$$

$$\text{Es folgt: } \hat{R} = \bigcap_{R' \in R_0} R' \subseteq R^*$$

Beweis der Hilfsbehauptungen:

(H1) Sei $a \in A$ beliebig

Wir wählen $n=0$ und damit die Kette (a, a)

\Rightarrow für beliebige $a \in A$ ist $(a, a) \in R^*$

(H2) Seien $a, b, c \in A$ mit $(a, b) \in R^*$ und $(b, c) \in R^*$

$(a, b) \in R^* \Rightarrow$ es gibt $a_0, \dots, a_n \in A$ mit $a = a_0, b = a_n$ und $(a_i, a_{i+1}) \in R^*$

$(b, c) \in R^* \Rightarrow$ es gibt $b_0, \dots, b_m \in A$ mit $b_0 = b, b_m = c$ und $(b_i, b_{i+1}) \in R^*$

Seien $c_0 = a_0, c_1 = a_1, \dots, c_n = a_n, c_{n+1} = b_1, \dots, c_{n+m} = b_m$

Es gilt $c_0 = a, c_{n+m} = c$ und $(c_i, c_{i+1}) \in R^*$ für $(0 \leq i < n+m)$

Also ist $(a, c) \in R^*$

(H3) Sei $(a, b) \in R$ beliebig

Wir wählen $n=1$ und die Kette a_0, a_1 mit $a_0 = a, a_1 = b$ und $(a_0, a_1) \in R$

Also gilt $(a, b) \in R^*$

3. Beweis zu $R^* \subseteq \hat{R}$:

Wir zeigen $R^* \subseteq R'$ für alle $R' \in R_0$

Daraus folgt $R^* \subseteq \bigcap_{R' \in R_0} R' = \hat{R}$

Sei R' beliebig

Sei $(a, b) \in R^*$

Hieraus folgt: es gibt $a_0, \dots, a_n \in A$ mit $a_0 = a, a_n = b, (a_i, a_{i+1}) \in R$ für $(0 \leq i < n)$

Wir zeigen: $(a_0, a_i) \in R'$ für alle $0 \leq i < n$

Beweis über vollständige Induktion über $0 \leq i < n$

I.A. $i = 0$: $(a_0, a_0) \in R'$ da R' reflexiv

I.S. $i \rightarrow i+1$

Induktionsannahme $(a_0, a_i) \in R'$

$(a_0, a_i) \in R', (a_i, a_{i+1}) \in R \subseteq R'$

Also: $(a_0, a_{i+1}) \in R'$

Für $i = n$: $(a_0, a_n) = (a, b) \in R'$ q.e.d.

Kapitelübersicht:**Kapitel 3 Mathematische Grundlagen**

3.1 Aussagenlogik

3.2 Prädikatenlogik

3.3 Mengen

3.4 Relationen

3.5 Folgen, Worte

3.6 Kontextfreie Grammatiken

3.5 Folgen, Worte

Sei A eine nichtleere Menge. Wir möchten Elemente aus A endlich aneinanderreihen.

Definition (endliche Menge):

Eine Folge der Länge n über A ist eine Funktion

$$a : [1, n] \longrightarrow A \quad n \in \mathbb{N}_0, [1, n] = \{1, 2, \dots, n\}$$

Die leere Folge $a : [1, 0] \longrightarrow A$ ist eindeutig bestimmt.

Meist wird sie mit ε bezeichnet.

$A^n = \{a \mid a : [1, n] \longrightarrow A\}$ Menge aller Folgen der Länge n über A

$A^* = \bigcup_{n \geq 0} A^n$ Menge aller Folgen über A

$A^+ = \bigcup_{n \geq 1} A^n$ Menge aller nichtleeren Folgen über A

$$A^0 = \{\Sigma\}$$

Operation conc: $A^* \times A^* \longrightarrow A^*$

conc(a, b) = c , seien $a : [1, n] \rightarrow A, b : [1, m] \rightarrow A$
mit $c : [1, n + m] \rightarrow A$

$$c(1) = a(1), c(2) = a(2), \dots \quad c(n) = a(n)$$

$$c(n + 1) = b(1), \dots \quad c(n + m) = b(m)$$

Uns interessieren Folgen von **Zeichen**.

Alphabet Σ nichtleere Menge

Definition (Wort):

Sei Σ Alphabet. Eine endliche Folge von Zeichen aus Σ heißt Wort.

$$\text{lexanal: } \Sigma^+ \rightarrow (\Sigma^+)^+$$

Schreibweise:

$a[1, n] \rightarrow \Sigma$ wird geschrieben als a_1, a_2, \dots, a_n $a_i = a(i)$

conc(a, b) geschrieben als $a.b$ oder ab

3.6 Kontextfreie Grammatiken

In diesem Abschnitt werden Mechanismen zur Beschreibung der Syntax von Programmiersprachen vorgestellt.

Beispiele sind die Sprache aussagenlogischer Terme, SML-Namen und einfache SML-Ausdrücke.

3.6.1 Einführendes Beispiel

Definition aussagenlogische Termine (Peter Mirolid):

1. w, f und jede aussagenlogische Variable sind Terme
2. Sind p und q Terme, dann auch $(\neg p), (p \wedge q), (p \vee q), (p \rightarrow q), (p \leftrightarrow q)$
3. nichts sonst

Formalisierung:

v stehe für aussagenlogische Terme

1. $\text{term} ::= w$
 $\text{term} ::= f$
 $\text{term} ::= v$
2. $\text{term} ::= (\neg \text{term})$
 $\text{term} ::= (\text{term} \wedge \text{term})$
 $\text{term} ::= (\text{term} \vee \text{term})$
 $\text{term} ::= (\text{term} \rightarrow \text{term})$
 $\text{term} ::= (\text{term} \leftrightarrow \text{term})$

Lesen:

- definierendes Vorkommen ($\text{term} ::=$): lesen als: „ein Term ist...“
- angewandtes Vorkommen (term rechts von $::=$): lesen als: „etwas, das ein Term ist...“

3.6.2 Definition kontextfreie Grammatiken

Welche Bestandteile hat die obige Beschreibung?

1. Welche Zeichen können in Termen auftreten?
 $w, f, v, (,), \neg, \wedge, \vee, \rightarrow, \leftrightarrow$ **Terminale**
2. term : **nichtterminal** (syntaktische Kategorie)
3. $\text{Nichtterm} ::=$ Wort aus Nichtterminalen und Terminalen **Produktionsregel**

Insgesamt:

Definition (kontextfreie Grammatik)

Eine kontextfreie Grammatik (kfG) ist ein Quadrupel $G = (N, \Sigma, P, S)$ mit

- N endliche, nichtleere Menge von **Nichtterminalen**.
- Σ nichtleeres Alphabet, **Terminale** $N \cap \Sigma = \emptyset$

- $P \subseteq (N \cup \Sigma)^*$
- $S \in N$ **Startsymbol**

Beispiel:

$$P = \left\{ \begin{array}{l} \text{id} ::= \text{symid} \\ \text{id} ::= \text{alphanumeric} \\ \text{alphanumeric} ::= ' \text{idrest} \\ \text{alphanumeric} ::= \text{b idrest} \\ \text{idrest} ::= \Sigma \\ \text{idrest} ::= \text{idrest b} \\ \text{idrest} ::= \text{idrest z} \\ \text{idrest} ::= \text{idrest } _ \\ \text{idrest} ::= \text{idrest '} \end{array} \right\}$$

$$\Sigma = \{b, z, ', _ \}$$

$$N = \{\text{id}, \text{symid}, \text{alphanumeric}, \text{idrest}\}$$

$$S = \text{id}$$

Bemerkung: die Regeln für symid fehlen

Ableitung: $((\neg(v \wedge v) \vee (v \rightarrow v))$

$$\begin{array}{l} \text{term} \Rightarrow (\text{term} \vee \text{term}) \Rightarrow ((\underbrace{\neg(\text{term})}_{\varphi}) \vee \underbrace{\text{term}}_{\psi}) \\ \Rightarrow ((\underbrace{\neg(\text{term} \wedge \text{term})}_{\varphi}) \vee \underbrace{\text{term}}_{\psi}) \end{array} \quad X ::= \alpha$$

generische Namen:

$$\begin{array}{l} a, b, c \in \Sigma \\ x, y, z \in \Sigma^* \\ \alpha, \beta, \gamma, \varphi, \psi \in (N \cup \Sigma)^* \\ A, B, \dots, X, Y, Z \in N \end{array}$$

Definition (leitet direkt ab)

Seien $\alpha, \beta \in (N \cup \Sigma)^*$

Wir schreiben $\alpha \Rightarrow_G \beta$ und sagen, β wird direkt aus α abgeleitet,

$$\begin{array}{l} \text{wenn es } \varphi, \psi, A, \gamma \text{ gibt} \quad \begin{array}{l} \alpha = \varphi A \psi \\ \beta = \varphi \gamma \psi \end{array} \\ \text{und} \quad A ::= \gamma \in P \end{array}$$

Sei \Rightarrow_G^* die reflexive transitive Hülle von \Rightarrow_G

Wir wählen durch die Definition „von unten“;

$$\alpha \Rightarrow_G^* \beta \text{ genau dann wenn es gibt } \alpha_0, \dots, \alpha_n \text{ mit } \alpha_0 = \alpha, \alpha_n = \beta \text{ und } \alpha_i \Rightarrow_G^* \alpha_{i+1} \text{ für alle } 0 \leq i < n$$

Solch eine Kette $\alpha_0, \dots, \alpha_n$ nennen wir eine **Ableitung** von β aus α .

$$\text{Gilt in jedem Schritt } \alpha_i \Rightarrow_G^* \alpha_{i+1} \quad ; \quad \alpha_i = \varphi_i A_i \psi_i \text{ und } \alpha_{i+1} = \varphi_i \gamma_i \psi_i \text{ mit } \varphi_i \in \Sigma^*$$

dann heißt die Ableitung **Linksableitung**.

Gilt $S \xRightarrow[G]{*} \alpha$, so heißt α **Satzform** von G .

Ist $\alpha \in \Sigma^*$, so heißt es ein **Satz** von G .

$L(G) = \left\{ w \in \Sigma^* \mid S \xRightarrow[G]{*} w \right\}$ **Sprache** von G

Beispiel: $L(G_{term})$ ist die Sprache der aussagenlogischen Terme über der Variablen v .

Definition (gerichteter Graph)

Ein **gerichteter Graph** ist ein Paar von Mengen (V, E) .

V heißt die **Knotenmenge**, $E \subseteq V \times V$ heißt die **Kantenmenge**. Der **Eingrad** $\text{indeg}(v)$ eines Knotens v ist $\text{indeg}(v) = |\{(w, v) \mid w \in V\}|$.

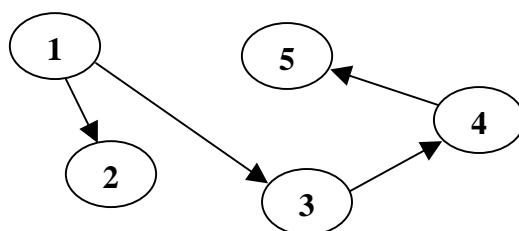
Der **Ausgrad** $\text{outdeg}(v) = |\{(v, w) \mid w \in V\}|$

Ein **Pfad** ist eine Folge von Knoten v_1, v_2, \dots, v_n mit $n \geq 1$ und $(v_i, v_{i+1}) \in E$ für $1 \leq i < n$.

Ein Graph (V, E) heißt **azyklisch** genau dann wenn es keinen Knoten mit einem Pfad zu sich selbst gibt, d.h. v_1, \dots, v_n mit $v_1 = v_n$.

Beispiel:

Sei $V = \{1, 2, 3, 4, 5\}$, $E = \{(1, 2), (1, 4), (3, 4), (4, 5), (5, 3)\}$



Dieser Graph hat z.B. den Pfad 1, 4, 5. Er hat keinen Zyklus

Definition (Baum)

Ein gerichteter azyklischer Graph (V, E) heißt **Baum**, genau dann wenn es einen Knoten mit Eingrad 0 gibt und alle anderen Knoten den Eingrad 1 haben.

Der Knoten mit Eingrad 0 heißt die **Wurzel** des Baumes.

Die Knoten mit Ausgrad 0 heißen **Blätter**. Nicht-Blattknoten heißen **innere Knoten**. Ist $((v, w) \in E)$ so heißt v **Elter** von w .

Ein Baum heißt **geordnet**, wenn es zu jedem Knoten mit Ausgrad k eine Funktion von $[1, k]$ auf die Kinder gibt.

Definition (beschrifteter Graph)

Ein Graph (V, E) heißt **knotenbeschriftet**, wenn es ein Alphabet a und eine Funktion $l: V \rightarrow a$ gibt.

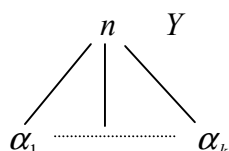
Ein Graph heißt **kantenbeschriftet**, wenn es a und $l: E \rightarrow a$.

Definition (Syntaxbaum)

Gegeben hfG $G = (N, \Sigma, P, S)$

Ein **Syntaxbaum** für $X \in N$ und $x \in \Sigma^*$ gemäß G ist ein geordneter, knotenbeschrifteter Baum mit

1. Wurzel ist beschriftet mit X
2. Das Blattwort ist x
3. Ist n ein innerer Knoten mit Beschriftung Y und sind die Kinder von links nach rechts beschriftet mit $\alpha_1, \dots, \alpha_k$, so gilt



$$\begin{aligned} Y &\in N \\ Y &::= X_1, \dots, X_k \in P \\ k = 0 : X &::= \varepsilon \end{aligned}$$

Sei $\alpha_0, \dots, \alpha_n$ eine Linksableitung von w aus X (d.h. $\alpha_0 = x, \alpha_n = w$)

Der zu dieser Ableitung korrespondierende Sytaxbaum wird wie folgt konstruiert:

1. Die Wurzel wird mit X beschriftet
2. Im Schritt $\varphi_i A_i \psi_i \Rightarrow_G \varphi_i \gamma_i \psi_i$ mit $\gamma_i = x_i, \dots, x_k$, $x_i \in (N \cup \Sigma)$

$k > 0$: füge k Knoten mit den Beschriftungen X_1, \dots, X_k (von links nach rechts) und k Kanten von dem am weitesten links stehenden mit A_i beschrifteten Knoten zu dem neuen Knoten hinzu.

$k = 0$: Füge einen Knoten, beschriftet mit ε und eine Kante von dem am weitesten links stehenden mit A_i beschrifteten Knoten zu dem neuen Knoten hinzu.

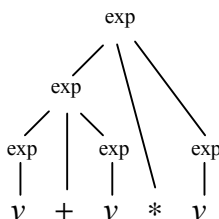
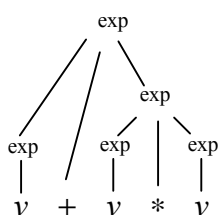
Definition (mehrdeutig):

Eine kfG G heißt **mehrdeutig**, genau dann wenn es ein $w \in L(G)$ mit zwei verschiedenen Syntaxbäumen gibt.

Beispiel: arithmetische Ausdrücke

$$G_m = (\{\text{exp}\}, \{+, *, \sim, v\}, P_m, \text{exp})$$

$$P_m = \left\{ \begin{array}{l} \text{exp} ::= \text{exp} + \text{exp} \\ \text{exp} ::= \text{exp} * \text{exp} \\ \text{exp} ::= \sim \text{exp} \\ \text{exp} ::= v \\ \text{exp} ::= (\text{exp}) \end{array} \right\}$$



G_m ist mehrdeutig.

Eindeutigkeit ist wichtig, weil die Semantik (Auswertung) gemäß der syntaktischen Struktur passiert. Verschiedene Syntaxbäume für den gleichen Satz führen im Allgemeinen zu verschiedenen Ergebnissen bei der Auswertung.

Satz: G_{term} ist Eindeutig

Beweis: Hilfsbehauptungen

(H1) Falls $w \in G_{term}$, dann gilt

Sei y ein Präfix von w (d.h. $w = yz$)

Def: $Ue(v) = |w|_v - |w|_v$

$|w|_a$ Anzahl der Vorkommen von a in w

$Ue(y) \geq 0$ und

$Ue(y) = 0$ genau dann wenn $y = \varepsilon$ oder $y = w$

Beweis: Induktion über die Länge von Ableitungen...

(H2) Sei $w \in L(G_{term})$ $|w| > 1$

Dann gibt es entweder einen eindeutig bestimmten Operator $op \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$ und eindeutig bestimmte $w_1, w_2 \in L(G_{term})$ mit $w = (w_1 \text{ op } w_2)$

oder es gibt ein eindeutig bestimmtes $w' \in L(G_{term})$ mit $w = (\neg w')$

Unvollständig geklammerte Ausdrücke, Operatoren mit **Präzedenzen**.

$e_1 \text{ op}_1 e_2 \text{ op}_2 e_3$ Annahme: $\text{Präz}(\text{op}_2) > \text{Präz}(\text{op}_1)$, dann Klammerung $e_1 \text{ op}_1 (e_2 \text{ op}_2 e_3)$

SML hat Präzedenzen für binäre infix-Operatoren:

Ebene 7: $/, *, \text{div}, \text{mod}$

Ebene 6: $+, -$

Ebene 4: $=, <, <=, >, >=$

Zusätzlich: unäre Operatoren (z.B. \sim) binden am stärksten. Funktionsanwendung bindet stärker als Stufe 7.

Stufe -1 andalso

Stufe -2 orelse

Assoziativität:

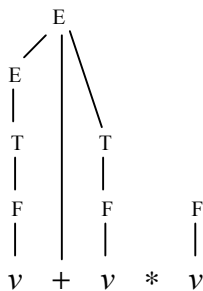
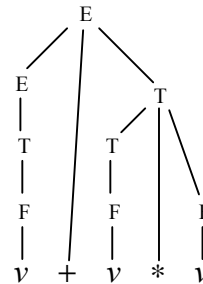
- linksassoziativ: klammert nach links
- rechtsassoziativ: klammert nach rechts
- arithmetische Operatoren klammern nach links

Arithmetische Ausdrücke in unvollständig geklammerten Notationen

Prinzip ein Nichtterminal pro Präzedenzstufe

$G_e = (\{E, T, F\}, \{+, *, \sim, (,), v\}, P_e, E)$

$$P_e = \left\{ \begin{array}{l} E ::= E + T \\ E ::= T \\ T ::= T * F \\ T ::= F \\ F ::= v \\ F ::= \sim F \\ F ::= (E) \end{array} \right.$$

Behauptung: G_e ist eindeutig $v + v * v$  $E * \text{nicht reduzierbar!}$ 

Einfache Ausdrücke im SML

```

exp ::=    infexp
          exp andalso exp
          exp orelse exp
          if exp then exp else exp
infexp ::= infexp id infexp
          appexp
appexp ::= atexp
atexp ::=  scon
          < op > vid
          (exp)
  
```

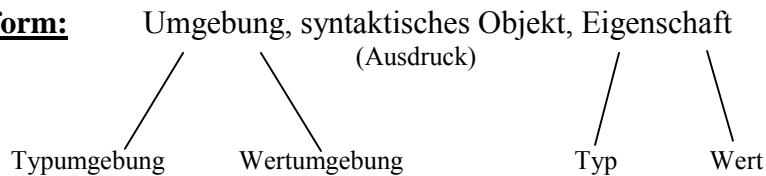
3.7 Inferenzsysteme

Inferenzsysteme werden benutzt zur Beschreibung von Typsystemen und von der Semantik.
Beispiele: einfache SML-Ausdrücke

Im Kapitel 1:

Beispiel für die Herleitung von Aussagen

- Typberechnung $\Gamma \vdash e : t$
“in der Typumgebung Γ hat der Ausdruck e den Typ t “
- Ausdrucksauswertung (Semantik) $\rho \vdash e \Downarrow v$
“in der Wertumgebung ρ hat e den Wert v “

Aussagenform:

Berechnungsregeln, damals informal beschrieben, werden formalisiert als **Inferenzsysteme**.

Allgemeine Form:

$$\frac{\text{Prämissen}}{\text{Konklusion}} \text{ Bedingung}$$

Prämisse, Konklusion sind Aussagen einer der obigen Formen.

3.7.1 Typberechnung

(Γ gegeben)

(intconst)

$$\frac{}{\Gamma \vdash id : \text{int}} a \text{ intconst}$$

(id)

$$\frac{}{\Gamma \vdash a : t} \Gamma(id) = (val, t)$$

(intplus)

$$\frac{\Gamma \vdash e_1 : \text{int} \quad \Gamma \vdash e_2 : \text{int}}{\Gamma \vdash (e_1 + e_2) : \text{int}}$$

(realplus)

$$\frac{\Gamma \vdash e_1 : \text{real} \quad \Gamma \vdash e_2 : \text{real}}{\Gamma \vdash (e_1 + e_2) : \text{real}}$$

(=)

$$\frac{\Gamma \vdash e_1 : t \quad \Gamma \vdash e_2 : t}{\Gamma \vdash (e_1 = e_2) : \text{bool}} \text{ "t hat Gleichheit"}$$

(andalso)

$$\frac{\vdash \quad \Gamma \vdash e_1 : \text{bool} \quad \Gamma \vdash e_2 : \text{bool}}{\vdash \quad \Gamma \vdash (e_1 \text{ andalso } e_2) : \text{bool}}$$

(if)

$$\frac{\Gamma \vdash e_1 : \text{bool} \quad \Gamma \vdash e_2 : t \quad \Gamma \vdash e_3 : t}{\Gamma \vdash (\text{if } e_1 \text{ then } e_2 \text{ else } e_3) : t}$$
Beobachtung über das Inferenzsystem:

Syntaktische Objekte in Prämissen sind die Bestandteile (Unterausdrücke) des syntaktischen Objekts der Konklusion.

Die Inferenzregel definiert die Berechnung der Eigenschaften des syntaktischen Objekts der Konklusion in Abhängigkeit von den Eigenschaften der syntaktischen Objekte der Prämissen.

Deshalb: **Syntaxgesteuertes Inferenzsystem**

Beispiel

$$\text{Gegeben } \Gamma = \begin{cases} a \mapsto (val, int) \\ b \mapsto (val, char) \\ c \mapsto (val, char) \end{cases}$$

Typberechnung für if $a = 0$ then b else c

$$\frac{\frac{\frac{}{\Gamma \vdash a : int} \Gamma \vdash (a) = (val, int) \quad \frac{}{\Gamma \vdash 0 : int} 0 \text{ intconst}}{\Gamma \vdash (a = 0) : bool} \quad \frac{\frac{}{\Gamma \vdash b : char} \Gamma \vdash (b) : (val, char) \quad \frac{}{\Gamma \vdash c : char} \Gamma \vdash (c) : (val, char)}{\Gamma \vdash (if (a = 0) then b else c) : char}$$

Baum wurde aufgebaut

- bei den Blättern angefangen
- bei der Wurzel aufgehört

Baumaufbau von der Wurzel zu den Blättern:

$$\frac{\frac{\frac{}{\Gamma \vdash a : t_2} \Gamma \vdash (a) = (val, int) \quad \frac{}{\Gamma \vdash 0 : t_2} 0 \text{ intconst}}{\Gamma \vdash (a = 0) : bool} \quad \frac{\frac{}{\Gamma \vdash b : t_1} \Gamma \vdash (b) : (val, char) \quad \frac{}{\Gamma \vdash c : t_1} \Gamma \vdash (c) : (val, char)}{\Gamma \vdash (if (a = 0) then b else c) : t_1} \quad \begin{matrix} t_1 = int \\ t_2 = int \end{matrix}$$

3.7.2 Auswertung (Semantik)

Sei Wertumgebung ρ gegeben

(intconst)

$$\frac{}{\rho \vdash a \Downarrow a} a \text{ intconst}$$

(id)

$$\frac{}{\rho \vdash id \Downarrow v} \rho(id) = (val, v)$$

(intplus)

$$\frac{\rho \vdash e_1 \Downarrow v_1 \quad \rho \vdash e_2 \Downarrow v_2}{\rho \vdash (e_1 + e_2) \Downarrow v_{int} \quad v_1 + v_2}$$

(=)

$$\frac{\rho \vdash e_1 \Downarrow v_1 \quad \rho \vdash e_2 \Downarrow v_2 \quad v_1 = v_2}{\rho \vdash (e_1 = e_2) \Downarrow true}$$

(=2)

$$\frac{\rho \vdash e_1 \Downarrow v_1 \quad \rho \vdash e_2 \Downarrow v_2 \quad v_1 \neq v_2}{\rho \vdash (e_1 = e_2) \Downarrow false}$$

(if₁)

$$\frac{\rho \vdash e_1 \Downarrow true \quad \rho \vdash e_2 \Downarrow v}{\rho \vdash (if e_1 then e_2 else e_3) \Downarrow v}$$

(if₂)

$$\frac{\rho \vdash e_1 \Downarrow false \quad \rho \vdash e_3 \Downarrow v}{\rho \vdash (if e_1 then e_2 else e_3) \Downarrow v}$$

Wichtig: Nur einer der beiden Ausdrücke e_2, e_3 wird ausgewertet.