Beispiele für Knuth-Bendix-V

- $E = \{ffg(x) = h(x), ff(x) = x, fh(x) = g(x)\} \quad >: KBO(3,2,1)$
  $R_0 = \emptyset, E_0 = E$
  $R_1 = \{ffg(x) \to h(x)\}, KP_1 = \emptyset. E_1 = \{ff(x) = x, fh(x) = g(x)\}$
  $R_2 = \{ffg(x) \to h(x), ff(x) \to x\}, NKP_2 = \{(g(x), h(x))\},$
  $E_2 = \{fh(x) = g(x), g(x) = h(x)\}, R_2 = \{ff(x) \to x\}$
  $R_3 = \{ff(x) \to x, fh(x) \to g(x)\}, NKP_3 = \{(h(x), fg(x))\}, E_3 =$
  $\{g(x) = h(x), h(x) = fg(x)\}$
  $R_4 = \{ff(x) \to x, fh(x) \to h(x), g(x) \to h(x)\}, NKP_3 = \emptyset, E_4 = \emptyset$
- $E = \{fgf(x) = gfg(x)\} \quad >: LL :: f > g$
  $R_0 = \emptyset, E_0 = E$
  $R_1 = \{fgf(x) \to gfg(x)\}, NKP_1 = \{(gfggf(x), fggfg(x))\}, E_1 =$
  $\{gfggf(x) = fggfg(x)\}$
  $R_1 = \{fgf(x) \to gfg(x), fggfg(x) \to gfggf(x)\}, NKP_2 =$
  $\{(gfggfggfg(x), fgggfggfg(x), ..\}...$

Verfeinertes Inferenzsystem

Definition 9.13 *Sei $>$ eine Noethersche PO auf $Term(F, V)$. Das Inferenzsystem $\mathcal{P}_{TES}$ besteht aus folgenden Regeln:*

(1) Orientieren
$$\frac{(E \cup \{s \doteq t\}, R)}{(E, R \cup \{s \to t\})} \text{ falls } s > t$$

(2) Generieren
$$\frac{(E, R)}{(E \cup \{s \doteq t\}, R)} \text{ falls } s \leftarrow_R \circ \to_R t$$

(3) Simplifizieren GL
$$\frac{(E \cup \{s \doteq t\}, R)}{(E \cup \{u \doteq t\}, R)} \text{ falls } s \to_R u$$

(4) Simplifizieren RS
$$\frac{(E, R \cup \{s \to t\})}{(E, R \cup \{s \to u)\})} \text{ falls } t \to_R u$$

(5) Simplifizieren LS
$$\frac{(E, R \cup \{s \to t\})}{(E \cup \{u \doteq t\}, R)} \text{ falls } s \to_R u \text{ mit } l \to r \text{ und } s \succ l \text{ (SubSumOrd.)}$$

(6) Löschen Identitäten

Gleichungsimplementierungen

Programmierung = Beschreibung von Algorithmen in einem Formalen System

Definition 10.1 *Sei $f : M_1 \times ... \times M_n \rightsquigarrow M_{n+1}$ eine (partielle) Funktion. Seien $T_i, 1 = 1...n+1$ entscheidbare Grundtermmengen über Σ, $\hat{f}$ n-stelliges Funktionssymbol, E Gleichungsmenge.*
Eine Deteninterpretation $\mathfrak{I}$ ist eine Funktion $\mathfrak{I} : T_i \rightarrow M_i$.

$\hat{f}$ implementiert f unter der Interpretation $\mathfrak{I}$ in E gdw
1) $\mathfrak{I}(T_i) = M_i (i = 1...n+1)$
2) $f(\mathfrak{I}(t_1), ..., \mathfrak{I}(t_n)) = \mathfrak{I}(t_{n+1})$ gdw $\hat{f}(t_1, ..., t_n) =_E t_{n+1}$ $(\forall t_i \in T_i)$

$$\begin{array}{ccc} T_1 \times ... \times T_n & \xrightarrow{\hat{f}} & T_{n+1} \\ \mathfrak{I} \downarrow \quad \mathfrak{I} \downarrow & & \mathfrak{I} \downarrow \\ M_1 \times ... \times M_n & \xrightarrow{f} & M_{n+1} \end{array}$$

Abkürzung: $(\hat{f}, E, \mathfrak{I})$ *implementiert* f.

Gleichungsimplementierungen

Satz 10.1 *E sei Gleichungs- oder Regelmenge. Für alle $i = 1, ..., n+1$ gelte*
1) $\mathfrak{I}(T_i) = M_i$
2a) $f(\mathfrak{I}(t_1), ..., \mathfrak{I}(t_1)) = \mathfrak{I}(t_{n+1}) \rightsquigarrow \hat{f}(t_1, ..., t_n) =_E t_{n+1}\ (\forall t_i \in T_i)$

$\hat{f}$ implementiert die totale Funktion f unter $\mathfrak{I}$ in E wenn eine der folgenden Bedingungen gilt:

a) $\forall t, t' \in T_{n+1} : t =_E t' \rightsquigarrow \mathfrak{I}(t) = \mathfrak{I}(t')$
b) E *konfluent und* $\forall t \in T_{n+1} : t \to_E t' \rightsquigarrow t' \in T_{n+1} \wedge \mathfrak{I}(t) = \mathfrak{I}(t')$
c) E *konfluent und* T_{n+1} *enthält nur E-irreduzible Terme.*

Anwendung: $(\hat{f}, E, \mathfrak{I})$ implementiere die totale Funktion f. Wird E um E_0 Erweitert unter Beibehaltung von $\mathfrak{I}$, so gelten weiterhin 1 und 2a. Ist eines der Kriterien a, b, c für $E \cup E_0$ erfüllt, so implementiert auch $(\hat{f}, E \cup E_0, \mathfrak{I})$ die Funktion f. Dies gilt insbesondere wenn $E \cup E_0$ konfluent ist und T_{n+1} enthält nur $E \cup E_0$ irreduzible Terme.

Gleichungsimplementierungen

Satz 10.2 *$(\hat{f}, E, \mathfrak{I})$ implementiere die (partielle) Funktion f. Dann gilt:*

a) $\forall t, t' \in T_{n+1} :: \mathfrak{I}(t) = \mathfrak{I}(t') \wedge \mathfrak{I}(t) \in Bild(f) \rightsquigarrow t =_E t'$
b) Sei E konfluent und T_{n+1} enthalte nur E Normalformen. Dann ist $\mathfrak{I}$ injektiv auf $\{t \in T_{n+1} : \mathfrak{I}(t) \in Bild(f)\}$.

Satz 10.3 *Kriterium zur Implementierung totaler Funktionen. Es gelte*

1) $\mathfrak{I}(T_i) = M_i \quad (i = 1, ..., n+1)$
2) $\forall t, t' \in T_{n+1} :: \mathfrak{I}(t) = \mathfrak{I}(t')$ *gdw* $t =_E t'$
3) $\forall_{1 \leq i \leq n} \quad t_i \in T_i \quad \exists t_{n+1} \in T_{n+1} ::$

$$\hat{f}(t_1, ..., t_n) =_E t_{n+1} \wedge f(\mathfrak{I}(t_1), ...\mathfrak{I}(t_n)) = \mathfrak{I}(t_{n+1})$$

Dann implementiert $\hat{f}$ die Funktion f unter $\mathfrak{I}$ in E und f ist total.

Beachte: Enthält T_{n+1} nur Normalformen und ist E konfluent, so ist 2) erfüllt, wenn $\mathfrak{I}$ auf T_{n+1} injektiv ist.

Gleichungsimplementierungen

Satz 10.4 *$(\hat{f}, E, \mathfrak{I})$ implementiere $f : M_1 \times ... \times M_n \to M_{n+1}$. Seien $S_i = \{t \in T_i :: \exists t_0 \in T_i : t \neq t_0, \mathfrak{I}(t) = \mathfrak{I}(t_0) \;\; t \stackrel{+}{\to}_E t_0\}$ rekursive Mengen.*
Dann implementiert $\hat{f}$ mit Termmengen $T_i' = T_i \backslash S_i$ ebenfalls f unter $\mathfrak{I}|_{T_i}$ in E.

Man kann also aus T_i Terme streichen die auf andere Terme aus T_i mit gleichem $\mathfrak{I}$-Wert reduzierbar sind. Erlaubt somit Einschränkung auf E-Normalformen.

Folgerung 10.1

- *Man kann Implementierungen Komponieren.*
- *Erweitert man E um E Folgerungsgleichungen, dann bleibt die Implementierungseigenshaft erhalten. Wichtig für die KB-Vervollständigung da nur E-Folgerungen hinzugenommen werden.*

Beispiele: Aussagenlogik, Natürliche Zahlen

Beispiel 10.1 *Konvention: Gleichungen legen die Signatur fest. Gelegentlich variable Stellenzahl und Overloading. Einsortig.*

Boolsche Algebra: Sei $M = \{true, false\}$ *mit* $\wedge, \vee, \neg, \supset, \ldots.$ *Konstanten* tt, ff *Termmenge Bool* $:= \{tt, ff\}$, $\mathfrak{I}(tt) = true, \mathfrak{I}(ff) = false$.
Strategie: Vermeide Regeln mit tt *oder* ff *als linke Seite. Nach Satz 10.1 c) kann man Gleichungen mit diesen Einschränkungen hinzunehmen ohne die Implementierungseigenschaft zu beeinflüßen, sofern Konfluenz erreicht wird. Betrachte folgende Regeln:*

(1) $cond(tt, x, y) \to x$ *(2)* $cond(ff, x, y) \to y$. *(Hilfsfunktion).*
(3) $x \; vel \; y \to cond(x, tt, yy)$
$E = \{(1), (2), (3)\}$ *ist konfluent. Es gilt:*
$tt \; vel \; y =_E cond(tt, tt, y) =_E tt$ *d.h.*
$(*_1)$ $tt \; vel \; y = tt$ *und* $(*_2)$ $x \; vel \; tt = cond(x, tt, tt)$
$x \; vel \; tt = tt$ *läßt sich nicht aus* E *ableiten.*
Dennoch implementiert vel die Funktion $\vee$ *mit diesem* E.

Beispiele: Aussagenlogik

Nach Satz 10.3 sind Bedingungen (1), (2), (3) zu zeigen:
$\forall t, t' \in \mathit{Bool}\ \exists \bar{t} \in \mathit{Bool} :: \mathfrak{I}(t) \vee \mathfrak{I}(t') = \mathfrak{I}(\bar{t}) \wedge t\ \mathit{vel}\ t' =_E \bar{t}$

Für $t = tt$ $(*_1)$ und $t = ff$ (2) da $ff\ \mathit{vel}\ t' \to_E \mathit{cond}(ff, tt, t') \to_E t'$
D.h. $x\ \mathit{vel}\ tt \neq_E tt$ aber $tt\ \mathit{vel}\ tt =_E tt$, $ff\ \mathit{vel}\ tt =_E tt$.

MC Carthy's Regeln für *cond*:

(1) $\mathit{cond}(tt, x, y) = x$ (2) $\mathit{cond}(ff, x, y) = y$ (*) $\mathit{cond}(x, tt, tt) = tt$
Beachte: Nicht identisch mit *cond* in Lisp. Unterschied: Reihenfolge der Auswertung. Betrachte
(**) $\mathit{cond}(x, \mathit{cond}(x, y, z), u) \to \mathit{cond}(x, y, u) \rightsquigarrow$
$E' = \{(1), (2), (3), (*), (**)\}$ ist terminierend und konfluent.
Vereinbarungen: In Gleichungsmengen sind stets (1), (2), (3) und
$x\ et\ y \to \mathit{cond}(x, y, ff)$ enthalten.
Schreibweise: $\mathit{cond}(x, y, z) :: [x \to y, z]$ bzw.
$[x \to y_1, x_2 \to y_2, ..., x_n \to y_n, z]$ für $[x \to [...]..., z]$

Beispiele: Semantische Argumente

Eigenschaften implementierender Funktionen:
$(vel, E, \mathfrak{I})$ implementiert $\vee$ von BOOL.

Beh: *vel* ist assoziativ auf *Bool*.
Z.Z. $\forall t_1, t_2, t_3 \in Bool : t_1\ vel\ (t_2\ vel\ t_3) =_E (t_1\ vel\ t_2)\ vel\ t_3$

Es gibt $t_1, t_2, T_1, T_2 \in Bool$ mit
$\mathfrak{I}(t_2) \vee \mathfrak{I}(t_3) = \mathfrak{I}(t)$ und $\mathfrak{I}(t_1) \vee \mathfrak{I}(t_2) = \mathfrak{I}(t')$ sowie
$\mathfrak{I}(t_1) \vee \mathfrak{I}(t) = \mathfrak{I}(T)$ und $\mathfrak{I}(t') \vee \mathfrak{I}(t_3) = \mathfrak{I}(T')$

Wegen der semantisch gültigen Assoziativität von $\vee$ gilt
$\mathfrak{I}(T) = \mathfrak{I}(t_1) \vee \mathfrak{I}(t_2) \vee \mathfrak{I}(t_3) = \mathfrak{I}(T')$

Da *vel* $\vee$ implementiert folgt nun:
$t_1\ vel\ (t_2\ vel\ t_3) =_E t_1\ vel\ t =_E T =_E T' =_E t'\ vel\ t_3 =_E (t_1\ vel\ t_2)\ vel\ t_3$

Beispiele: Natürliche Zahlen

Funktionssymbole: $\hat{0}, \hat{s}$ Grundterme: $\{\hat{s}^n(\hat{0})\ (n \geq 0)\}$
$\mathfrak{I}$ Interpretation $\mathfrak{I}(\hat{0}) = 0, \mathfrak{I}(\hat{s}) = \lambda x.x + 1$, d.h. $\mathfrak{I}(\hat{s}^n(\hat{0})) = n\ (n \geq 0)$.
Abkürzung: $n \hat{+} 1 := \hat{s}(\hat{n})\ (n \geq 0)$
Zahlenterme. $NAT = \{\hat{n} : n \geq 0\}$ Normalformen (Satz 10.1 c gilt).

Wichtige Hilfsfunktionen auf NAT:

Sei $E = \{is\_null(\hat{0}) \rightarrow\ tt, is\_null(\hat{s}(x)) \rightarrow\ ff\}$.
is_null implementiert das Prädikat $Is\_Null : \mathbb{N} \rightarrow \{true, false\}$ Nulltest.
Erweitere E mit (nicht terminierende Regeln)
$\hat{g}(x) \rightarrow [is\_null(x) \rightarrow \hat{0}, \hat{g}(x)], \hat{f}(x) \rightarrow [is\_null(x) \rightarrow \hat{g}(x), \hat{0}]$

Beh: Unter der Standardinterpretation $\mathfrak{I}$ gilt:
$\hat{f}$ implementiert die Nullfunktion $f(x) = 0\ \ (x \in \mathbb{N})$ und
$\hat{g}$ implementiert die Funktion $g(0) = 0$ sonst undefiniert.
Wegen $\hat{f}(\hat{0}) \rightarrow [is\_null(\hat{0}) \rightarrow \hat{g}(\hat{0}), \hat{0}] \stackrel{*}{\rightarrow} \hat{g}(\hat{0}) \rightarrow [...] \stackrel{*}{\rightarrow} \hat{0}$ und
$\hat{f}(\hat{s}(x)) \rightarrow [is\_null(\hat{s}(x)) \rightarrow \hat{g}(\hat{s}(x))] \stackrel{*}{\rightarrow} \hat{0}$ folgt dies aus Satz 10.3.