Implementierung normierter Register Maschinen

Lemma 10.1 *M_e kann durch Gleichungssystem implementiert werden.*

Beweis: Sei tup_n n-stelliges Funktionssymbol. Für $t_i \in \mathbb{N}$ $(0 < i \leq n)$ sei $\langle t_1, ..., t_n \rangle$ Interpretation von $tup_n(\hat{t}_1, ..., \hat{t}_n)$. Programmterme werden durch sich selbst interpretiert (es sind ja Terme). Für $m \geq n$ ::

$$\begin{array}{ccc} P_n & tup_m(\hat{t}_1, ..., \hat{t}_m) & \text{Syntaktische Ebene} \\ \mathfrak{I} \downarrow & \mathfrak{I} \downarrow & \\ P_n & \langle t_1, ..., t_m \rangle & \text{Interpretation} \end{array}$$

Sei *eval* 2-stelliges Funktionssymbol zur Implementierung von M_e und $i \leq n$. Definiere $E_n := \{$

$$eval(a_i,\ tup_n(x_1, ..., x_n) \rightarrow\ tup_n(x_1, ..., x_{i-1}, \hat{s}(x_i), x_{i+1}, ..., x_n)$$
$$eval(s_i,\ tup_n(..., x_{i-1}, \hat{0}, x_{i+1}...) \rightarrow\ tup_n(..., x_{i-1}, \hat{0}, x_{i+1}...)$$
$$eval(s_i,\ tup_n(..., x_{i-1}, \hat{s}(x), x_{i+1}...) \rightarrow\ tup_n(..., x_{i-1}, x, x_{i+1}...)$$
$$eval(x_1x_2, t) \rightarrow\ eval(x_2,\ eval(x_1, t))$$
$$eval((x)_i,\ tup_n(..., x_{i-1}, \hat{0}, x_{i+1}...) \rightarrow\ tup_n(..., x_{i-1}, \hat{0}, x_{i+1}...)$$
$$eval((x)_i,\ tup_n(..., x_{i-1}, \hat{s}(y), x_{i+1}...) \rightarrow$$
$$eval((x)_i, eval(x,\ tup_n(..., x_{i-1}, \hat{s}(y), x_{i+1}...))\}$$

$(eval, E_n, \mathfrak{I})$ implementiert M_e

Betrachte Programmterme die höchstens Register mit $1 \leq i \leq n$ enthalten.

- E_n ist konfluent (links-linear, ohne kritische Paare).
- Satz 10.3 nicht anwendbar, da M_e nicht total. Bedingungen der Definition 10.1 überprüfen.

(1) $\mathfrak{I}(T_i) = M_i$ nach Definition.
(2) $M_e(p, \langle k_1, ..., k_n \rangle) = \langle m_1, ..., m_n \rangle$ gdw
$eval(p,\ tup_n(\hat{k}_1, ..., \hat{k}_n)) =_{E_n} tup_n(\hat{m}_1, ..., \hat{m}_n)$
$\curvearrowright$ Aus der Def. von M_e bzw. E_n. Induktion Aufbau von p.
$\curvearrowleft$ Induktion über Aufbau von p ::
1. $p = a_i(s_i)$ ::$\hat{k}_j = \hat{m}_j (j \neq i), \hat{s}(\hat{k}_i) = \hat{m}_i$ bzw. $\hat{k}_i = \hat{m}_i = \hat{0}$
$(\hat{k}_i = \hat{s}(\hat{m}_i))$ für s_i
2.Sei $p = p_1 p_2$ und
$eval(p_2, eval(p_1,\ tup_n(\hat{k}_1, ..., \hat{k}_n))) \xrightarrow{*}_{E_n} tup_n(\hat{m}_1, ..., \hat{m}_n)$
Wegen der Regeln in E_n gilt:

$(eval, E_n, \mathfrak{I})$ implementiert M_e

Es gibt $i_1, ..., i_n \in \mathbb{N}$ mit $eval(p_1, tup_n(\hat{k}_1, ..., \hat{k}_n)) \xrightarrow{*}_{E_n} tup_n(\hat{i}_1, ..., \hat{i}_n)$
also auch
$eval(p_2, tup_n(\hat{i}_1, ..., \hat{i}_n)) \xrightarrow{*}_{E_n} tup_n(\hat{m}_1, ..., \hat{m}_n)$
Nach Induktion Voraussetzung (2-Mal) gilt die Behauptung.
3. Sei $p = (p_1)_i$. Es gilt:
$eval((p_1)_i, tup_n(\hat{k}_1, ..., \hat{k}_n)) \xrightarrow{*}_{E_n} tup_n(\hat{m}_1, ..., \hat{m}_n)$
Es gibt endliche Folge $(t_j)_{1 \le j \le l}$ mit
$t_1 = eval((p_1)_i, tup_n(\hat{k}_1, ..., \hat{k}_n)), \quad t_j \to t_{j+1}, \quad t_l = tup_n(\hat{m}_1, ..., \hat{m}_n)$
Es gibt eine Teilfolge $(T_j)_{1 \le j \le m}$ der Gestalt $eval((p_1)_i, tup_n(\hat{i}_{1,j}, ..., \hat{i}_{n,j}))$
In T_m ist $i_{i,m} = 0$, d.h. $i_{1,m} = m_1, ..., i_{i,m} = 0 = m_i, ..., i_{n,m} = m_n$.
Für $j < m$ gilt stets $i_{i,j} \neq 0$ und
$eval(p_1, tup_n(\hat{i}_{1,j}, ..., \hat{i}_{n,j}) \xrightarrow{*}_{E_n} tup_n(\hat{i}_{1,j+1}, ..., \hat{i}_{n,j+1})$. Induktion Voraussetzung liefert:
$M_e(p_1, \langle i_{1,j}, ..., i_{n,j} \rangle) = \langle i_{1,j+1}, ..., i_{n,j+1} \rangle$ für $j = 1, ..., m$. Dann aber
$M_e((p_1)_i, \langle i_{1,j}, ..., i_{n,j} \rangle) = \langle m_1, ..., m_n \rangle \quad (1 \le j < m)$

Implementierung von $\mathfrak{R}_p$

Für $f \in \mathfrak{R}_p^{n,1}$ gibt es $r \in \mathbb{N}$, Programmterm p mit höchstens r-Registern $(n+1 \leq r)$, so dass für alle $k_1, ..., k_n, k \in \mathbb{N}$ gilt:
$f(k_1, ..., k_n) = k$ gdw $\forall m \geq 0$

$$eval(p, tup_{r+m}(\hat{k}_1, ..., \hat{k}_n, \hat{0}, \hat{0}, ..., \hat{0}, \hat{x}_1, ..., \hat{x}_m)) =_{E_{r+m}}$$
$$tup_{r+m}(\hat{k}_1, ..., \hat{k}_n, \hat{k}, \hat{0}, ..., \hat{0}, \hat{x}_1, ..., \hat{x}_m) \quad \text{gdw}$$

$$eval(p, tup_r(\hat{k}_1, ..., \hat{k}_n, \hat{0}, \hat{0}, ..., \hat{0})) =_{E_r} tup_r(\hat{k}_1, ..., \hat{k}_n, \hat{k}, \hat{0}, ..., \hat{0})$$

Beachte: $E_r \sqsubset E_{r+m}$ via $tup_r(...) \blacktriangleright tup_{r+m}(..., \hat{0}, ..., \hat{0})$.

Seien $\hat{f}, \hat{R}$ neue Funktionssymbole, p Programm für f. Erweitere E_r um $\hat{f}(y_1, ..., y_n) \to \hat{R}(eval(p, tup_r(y_1, ..., y_n), \hat{0}, ..., \hat{0}))$ und $\hat{R}(tup_r(y_1, ..., y_r)) = y_{n+1}$ zu $E_{ext(f)}$.

Satz 10.6 *$f \in \mathfrak{R}_p^{n,1}$ wird von $(\hat{f}, E_{ext(f)}, \mathfrak{I})$ implementiert.*

Nicht berechenbare Funktionen

Sei E rekursiv, T_i rekursiv. Dann ist das Prädikat

$$P(t_1, ..., t_n, t_{n+1}) \text{ gdw } \hat{f}(t_1, ..., t_n) =_E t_{n+1}$$

r.a. Prädikat auf $T_1 \times ... \times T_n \times T_{n+1}$
Implementiert $\hat{f}$ die Funktion f, so stellt P den Graphen der Funktion f dar $\rightsquigarrow$ $f \in \mathfrak{R}_p$.
Kleenescher Normalformsatz: $f(x_1, ..., x_n) = U(\mu_y[T_n(p, x_1, ..., x_n, y) = 0])$

Sei h totale nicht rekursive Funktion definiert durch:

$$h(x) = \begin{cases} \mu_y[T_1(x, x, y) = 0] & \text{falls } \exists y : T_1(x, x, y) = 0 \\ 0 & \text{sonst} \end{cases}$$

h wird eindeutig durch folgende Prädikate festgelegt:
(1) $(T_1(x, x, y) = 0 \wedge \forall z(z < y \rightsquigarrow T_1(x, x, z) \neq 0)) \rightsquigarrow h(x) = y$
(2) $(\forall z(z < y \wedge T_1(x, x, z) \neq 0)) \rightsquigarrow (h(x) = 0 \vee h(x) \geq y)$
Ersetzt man $h(x)$ durch u, so sind dies prim. rek. Prädikate in x, y, u.

Nicht berechenbare Funktionen

Es gibt primitiv rekursive Funktionen P_1, P_2 in x, y, u, so dass

$$(1') \quad P_1(x, y, h(x)) = 0 \text{ und } (2') \quad P_2(x, y, h(x)) = 0$$

(1) und (2) darstellen.
Es gibt Gleichungssystem E und Funktionssymbole $\hat{P}_1, \hat{P}_2$ die P_1, P_2 unter der standard Interpretation implementieren.
(Als prim. rek. Funktionen in den Var. x, y, u)
Sei $\hat{h}$ frisch. Füge zu E die Gleichungen

$$\hat{P}_1(x, y, \hat{h}(x)) = \hat{0} \quad \text{und} \quad \hat{P}_2(x, y, \hat{h}(x)) = \hat{0}$$

hinzu. Das Gleichungssystem ist Konsistent (es gibt Modelle) und $\hat{h}$ wird durch die Funktion h auf den natürlichen Zahlen interpretiert.$\rightsquigarrow$
Man kann also mit einer endlichen Gleichungsmenge implizit nicht rekursive Funktionen spezifizieren, wenn man beliebige Modelle als Interpretationen zulässt.
Durch nicht rekursive Gleichungsmengen kann man beliebige Funktionen als Interpretation eines konfluenten, terminierendes Grundsystem erhalten: $E = \{\hat{h}(\hat{t}) = \hat{t}' : t, t' \in \mathbb{N}, h(t) = t'\}$ (Regelanwendung nicht effektiv).

Berechenbare Algebren

Definition 10.3

- *Eine sig-Algebra $\mathfrak{A}$ ist rekursiv (effektiv, berechenbar), falls ihre Trägermengen rekursiv und alle Operationen rekursive Funktionen sind.*
- *Eine Spezifikation $spec = (sig, E)$ ist rekursiv, falls T_{spec} rekursiv ist.*

Beispiel 10.2 *Sei $sig = (\{nat, gerade\}, odd :\rightarrow gerade, 0 :\rightarrow nat, s : nat \rightarrow nat, red : nat \rightarrow gerade)$.*
Als sig-Algebra $\mathfrak{A}$ wähle: $A_{gerade} = \{2n : n \in \mathbb{N}\} \cup \{1\}, A_{nat} = \mathbb{N}$ mit

$$red \text{ als } \lambda x.\, if\ x \text{ gerade then } x \text{ else } 1, \qquad s \text{ Nachfolger}$$

Behauptung: Es gibt keine endliche (init-Algebra) Spezifikation für $\mathfrak{A}$

- *Keine Gleichungen der Sorte nat.*
- *$odd, red(s^n(0)), red(s^n(x))$ $(n \geq 0)$ Terme der Sorte gerade. Keine Gleichungen der Form $red(s^n(x)) = red(s^m(x)$ $(n \neq m)$ möglich.*
- *Unendlich viele Grundgleichungen nötig.*

Berechenbare Algebren

Lösung: Anreicherung der Signatur mit:
$gerade : nat \to nat$ und $cond : nat\ nat\ nat \to nat$ mit Interpretation

$$\lambda x.\ if\ x\ gerade\ then\ 0\ else\ 1, \qquad \lambda x, y, z.\ if\ x = 0\ then\ y\ else\ z$$

Gleichungen:
$gerade(0) = 0, gerade(s(0)) = s(0), gerade(s(s(x)) = gerade(x)$
$cond(0, y, z) = y, cond(s(x), y, z) = z$
$red(x) = cond(gerade(x), red(x), odd)$

Alternative: Bedingte Gleichungen:
$red(s(0)) = odd,\ red(s(s(x)) = odd\ if\ red(x) = odd$

Bedingte Gleichungssysteme (Termersetzungssysteme) sind offenbar Ausdruckstärker als reine Gleichungssysteme. Sie definieren ebenfalls Reduktionsrelationen. Konfluenz- und Terminierungskriterien lassen sich angeben. Negative Gleichungen in den Bedingungen führen zu Problemen mit der Initialen Semantik (keine Horn-Klausel Spezifikationen).

Berechenbare Algebren: Ergebnisse

Satz 10.7 *Sei $\mathfrak{A}$ eine rekursive termerzeugte sig- Algebra. Dann gibt es eine endliche Anreicherung sig' von sig und eine endliche Spezifikation $spec' = (sig', E)$ mit $T_{spec'}|_{sig} \cong \mathfrak{A}$.*

Satz 10.8 *Sei $\mathfrak{A}$ eine termerzeugte sig- Algebra. Dann sind äquivalent:*

- *$\mathfrak{A}$ ist rekursiv.*
- *Es gibt eine endliche Anreicherung (ohne neue Sorten) sig' von sig und ein endliches konvergentes Regelsystem R, so dass $\mathfrak{A} \cong T_{spec'}|_{sig}$ für $spec' = (sig', R)$*

Siehe Bergstra, Tucker: Characterization of Computable Data Types (Math. Center Amsterdam 79).

Achtung: Gilt nicht wenn man sich nur auf einstellige Funktionssymbole einschränkt.

Reduktionsstrategien für Ersetzungssysteme

Grundlegende Implementierungsprobleme für funktionale Programmiersprachen.
Welche Reduktionsstrategien garantieren die Berechnung von Normalformen falls diese existieren. Sei R TES, $t \in Term(\Sigma)$.
Ang. es gibt $\bar{t}$ irreduzibel mit $t \rightarrow_R^* \bar{t}$.

- Welche Auswahl der Redexe garantiert eine "Berechnung" von $\bar{t}$.
- Welche Auswahl der Redexe liefern "kürzeste" Ableitungsketten.
- Sei R terminierend. Gibt es eine Reduktionsstrategie die stets die kürzesten Ableitungsketten liefert. Was kostet sie?

Für $SKI-$Kalkül und $\lambda-$Kalkül ist die Left-Most-Outermost Strategie (normale Strategie) normalisierend, d.h. sie berechnet eine Normalform eines Terms wenn sie existiert. Sie liefert nicht die kürzesten Ableitungsketten. Es gilt jedoch: Ist $t \stackrel{k}{\rightarrow} \bar{t}$ eine kürzeste Ableitungskette, so $t \rightarrow_{LMOM}^{\leq 2^k} \bar{t}$. Durch Structure-Sharing-Methoden kann die Schranke für LMOM kleiner gewählt werden.

Reduktionsstrategien für Ersetzungssysteme

Definition 11.1 *Sei R ein TES.*

- *Eine Einschritt-Reduktionsstrategie $\mathfrak{S}$ für R ist eine Abbildung $\mathfrak{S} : Term(R, V) \to Term(R, V)$ mit $t = \mathfrak{S}(t)$ falls t in Normalform und $t \to_R \mathfrak{S}(t)$ sonst.*
- *$\mathfrak{S}$ ist eine Mehrschritt-Reduktionsstrategie für R wenn $t = \mathfrak{S}(t)$ falls t in Normalform und $t \overset{+}{\to}_R \mathfrak{S}(t)$ sonst.*
- *Eine Reduktionsstrategie $\mathfrak{S}$ heißt normalisierend für R, falls für jeden Term t der eine Normalform hat die Folge $(\mathfrak{S}^n(t))_{n\geq 0}$ eine Normalform enthält. (Insbesondere endlich ist).*
- *Eine Reduktionsstrategie $\mathfrak{S}$ heißt cofinal für R, falls für jedes t und $r \in \Delta^*(t)$ es ein $n \in \mathbb{N}$ gibt mit $r \overset{*}{\to}_R \mathfrak{S}^n(t)$.*

Cofinale Reduktionsstrategien sind die Besten im folgenden Sinn: Sie liefern maximalen Informationsgewinn.
Normalformen haben stets maximale Information.

Bekannte Reduktionsstrategien

Definition 11.2 *Reduktionsstrategien:*

- *Leftmost-Innermost (Call-by-Value). Einschritt-RS, reduziert wird Redex der am weitesten links im Term vorkommt und keinen echten Redex enthält.*
- *Paralell-Innermost. Mehrschritt-RS.* $PI(t) = \overline{t}$, *wobei* $t \mapsto \overline{t}$ *(Alle innermost Redexe werden reduziert).*
- *Leftmost-Outermost (Call-by-Name). Einschritt-RS.*
- *Parallel-Outermost. Mehrschritt-RS.* $PO(t) = \overline{t}$, *wobei* $t \mapsto \overline{t}$ *(Alle disjunkte outermost Redexe werden reduziert).*
- *Fair-LMOM. Ein Left-Most Outermost Redex in einer Red-Folge wird irgendwann reduziert. (Ein LMOR bleibt bei einer solchen Strategie nicht unreduziert). (Lazy Strategie).*

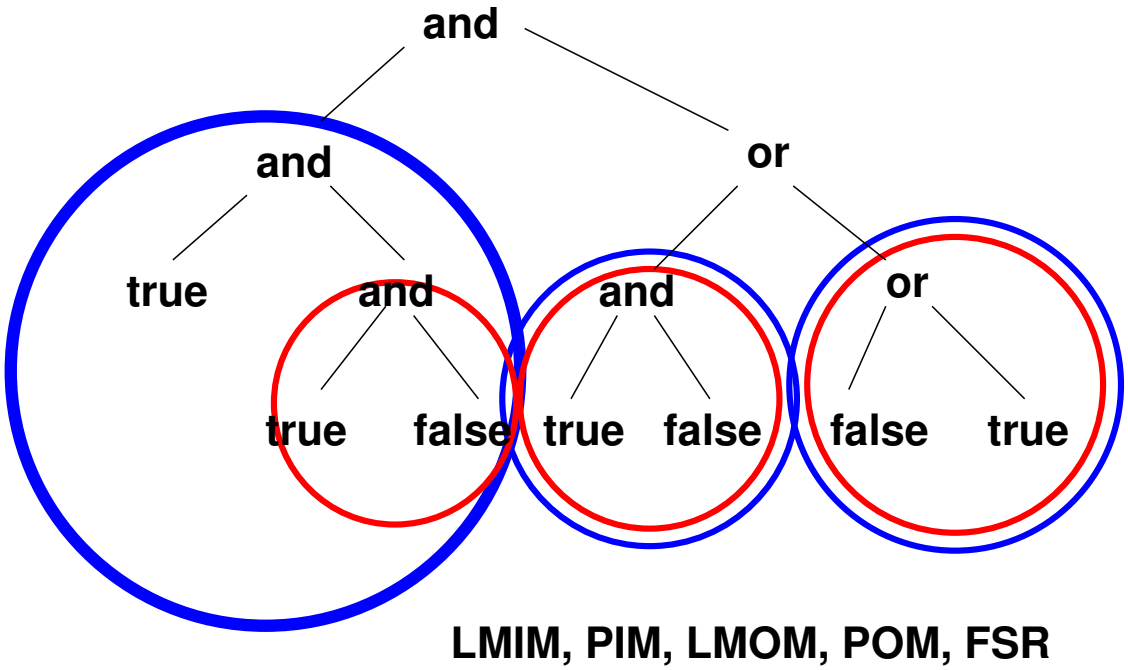
Bekannte Reduktionsstrategien

- Full-Substitution-rule. (Nur für orthogonale Systeme).
  Mehrschritt-RS. $GK(t) :: t \xrightarrow{+} GK(t)$ alle Redexe in t werden reduziert, falls nicht Disjunkt, so werden die Reste der Redexe danach noch reduziert.
- Call-By-Need. Einschritt-RS. Es wird stets ein notwendiger Redex reduziert. Ein Redex in t ist notwendig, wenn er zur Berechnung der Normalform reduziert werden muss. (Nur für bestimmte TES z.B. LMOM für SKI Kalkül)
  Problem: Wie entscheidet man ob ein Redex notwendig ist.
- Variable-Delay-Strategy: Einschritt-RS. Reduziere Redex der nicht als Redex in Variableninstanz eines anderen Redexes vorkommt.

Beispiele

Beispiel 11.1 *:*

- $and(true, x) \to x$, $and(false, x) \to false$,
  $or(true, x) \to true$, $or(false, x) \to x$
  Orthogonal, stark links sequentiell (Konstanten vor Variablen).



Beispiele

- $\Sigma = \{0, s, p, if0, F\}, R = \{p(0) \to 0, p(s(x)) \to x, if0(0, x, y) \to x, if0(s(z), x, y) \to y, F(x, y) \to if0(x, 0, F(p(x), F(x, y)))\}$
  Links -linear, ohne Überlappungen.(Orthogonal).
  $F(0,0) \to if0(0, 0, F(p(0), F(0,0))) \overset{OM}{\to} 0$
  $\downarrow PIM$
  $if0(0, 0, F(0, if0(0, 0, F(p(0), F(0,0)))))$
  Keine IM-Strategie ist für beliebige orthogonale Systeme normalisierend, erst recht nicht cofinal.
- FSR (Full-Substitution-Rule): Wähle alle Redexe im Term und reduziere sie von innermost zu outermost (beachte Redexe werden nicht zerstört). Cofinal für orthogonale Systeme.
- $\Sigma = \{a, b, c, d_i : i \in \mathbb{N}\}$
  $R := \{a \to b, d_k(x) \to d_{k+1}(x), c(d_k(b)) \to b$
  konfluent (links linear parallel 0-abgeschlossen).
  $c(d_0(a)) \to_1 c(d_1(a)) \to_1 \ldots.$ nicht normalisierend (POM).
  $c(d_0(a)) \to_{1,1} c(d_0(b)) \to_0 b$

Beispiele

- $\Sigma = \{a, b_i, c, d : i \in \mathbb{N}\}$. Nicht konfluentes WES:
  $R = \{ab_0c \to acb_0, ab_0d \to ad, c \to d, cb_i \to d, b_i \to b_{i+1}(i \geq 1)\}$
  $ab_0c \to_{11} ab_0d \to ad$
  $ab_0c \to_0 acb_0 \to_{11} acb_1 \to adb_1 \to ...$
- $\Sigma = \{f, a, b, c, d\}$ $R = \{f(x, b) \to d, a \to b, c \to c\}$ Orthogonal.
  LMOM muss nicht normalisierend sein:
  $f(c, a) \to f(c, a) \to ....$ aber $f(c, a) \to f(c, b) \to d$
- $f(a, f(x, y)) \to f(x, f(x, f(b, b)))$ links linear mit Überlappungen.
  $f(a, f(a, f(b, b))) \to_{OUT} f(a, f(a, f(b, b))) \to_{OUT} ....$
  $\downarrow_{INN}$
  $f(a, f(b, f(b, f(b, b)))) \to f(b, f(b, f(b, b)))$
- $R = \{f(g(x), c) \to h(x, d), b \to c\}$
  $f(g(f(a, f(a, \underline{b}))), c) \to_{VD} h(f(a, f(a, \underline{b})), d) \to_{VD}$
  $h(f(a, f(a, c)), d)$

Strategien für orthogonale Systeme

Satz 11.1 *Für orthogonale Systeme gilt:*

- *Full-Substitution-Rule ist eine cofinale Reduktionstrategie.*
- *POM ist eine normalisierende Reduktionstrategie.*
- *LMOM ist normalisierend für λ-Kalkül und CL-Kalkül.*
- *Jede Strategie die Fair-Outermost ist, ist normalisierend.*

Hilfmittel: Elementare Reduktionsdiagramme:

$$\begin{array}{ccccc} Sab(Ic) & \to & a(Ic)(b(Ic)) & \quad Ka(Ib) \to & Kab \\ \downarrow & & \downarrow & \quad \downarrow & \downarrow \\ & & ac(b(Ic)) & & \\ & & \downarrow & & \\ Sabc & \to & ac(bc) & \quad a \to_{\emptyset} & a \end{array}$$

$$\begin{array}{ccccccc} Ia & \to & a & Ia & \to & a & a \to a \\ \downarrow_{\emptyset} & & \downarrow_{\emptyset} & \downarrow & & \downarrow_{\emptyset} & \\ Ia & \to & a & a & \to_{\emptyset} & a & a \quad a \end{array}$$

Strategien für orthogonale Systeme

Projektionen:

$$
\begin{array}{ccccccc}
t_0 & \longrightarrow & t_1 & \longrightarrow & \ldots & \longrightarrow & t_n \\
\downarrow & & \downarrow & \longrightarrow & \longrightarrow & & \\
t'_2 & \longrightarrow & & \longrightarrow & \longrightarrow & & \\
\downarrow & & \downarrow & \longrightarrow & \longrightarrow & & \\
\downarrow & & \downarrow & \longrightarrow & \longrightarrow & & \\
\vdots & \vdots & \ldots & \ldots & & & \\
\downarrow & & \downarrow & \longrightarrow & \longrightarrow & & \\
t'_m & \longrightarrow & \longrightarrow & & & \longrightarrow &
\end{array}
$$

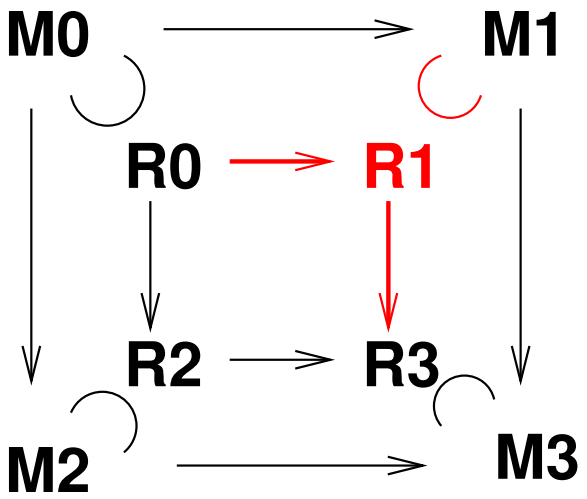
$R_4 = R_2 / R_1$

$R_3 = R_1 / R_2$

Seien $R_1 :: t \xrightarrow{+} t'$ und $R_2 :: t \xrightarrow{+} t'$ zwei Reduktionsfolgen von t nach t'.
Sie sind äquivalent $R_1 \cong R_2$ gdw $R_1 / R_2 = R_2 / R_1 = \emptyset$.

Strategien für orthogonale Systeme

Lemma 11.1 *Sei D ein elementares Reduktionsdiagramm für orthogonale Systeme, $R_i \subseteq M_i$ $(i = 0, 2, 3)$ Redexe mit $R_0 \to . \to . \to R_2 \to . \to . \to R_3$ d.h R_2 ist Rest von R_0 und R_3 ist Rest von R_2. Dann gibt es einen eindeutigen Redex $R_1 \subseteq M_1$ mit $R_0 \to . \to . \to R_1 \to . \to . \to R_3$, d-h.*



Strategien für orthogonale Systeme

Definition 11.3 *Sei* Π *ein Prädikat auf Termpaaren* M, R *so dass* $R \subseteq M$ *und* R *ist Redex (z.B. LMOM, LMIM,...).*

i) Π *hat Eigenschaft I wenn für das Vorliegen von* D *wie im Lemma gilt* $\Pi(M_0, R_0) \wedge \Pi(M_2, R_2) \wedge \Pi(M_3, R_3) \rightsquigarrow \Pi(M_1, R_1)$

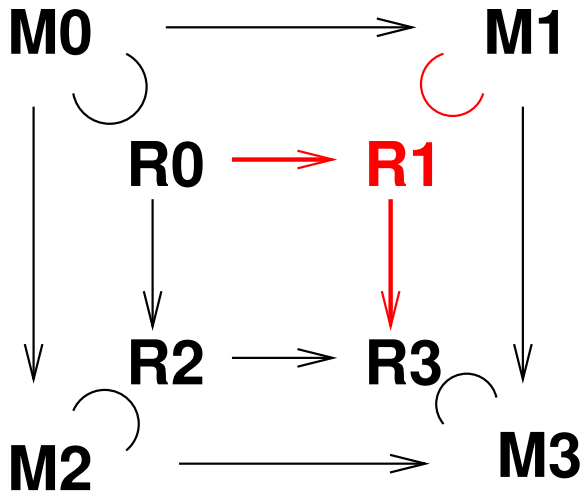
ii) Π *hat Eigenschaft II falls in jedem Reduktionsschritt* $M \rightarrow^R M'$ *mit* $\neg\Pi(M, R)$*, jeder Redex* $S' \subseteq M'$ *mit* $\Pi(M', S')$ *einen Vorgänger-Redex* $S \subseteq M$ *mit* $\Pi(M, S)$ *hat.*
(D.h. $\neg\Pi$ *Schritte produzieren keine neuen* Π*-Redexe).*

Lemma 11.2 *Trennbarkeit von Entwicklungen.* Π *habe Eigenschaft II. dann kann jede Entwicklung* $R :: M_0 \rightarrow \ldots \rightarrow M_n$ *aufgeteilt werden in einen* Π*-Anteil gefolgt von einem* $\neg\Pi$*-Anteil.D.h. es gibt Reduktionsfolgen* $R_\Pi :: M_0 = N_0 \rightarrow^{R_0} \ldots \rightarrow^{R_{k-1}} N_k$ *mit* $\Pi(N_i, R_i)$ $(i < k)$ *und* $R_{\neg\Pi} :: N_k \rightarrow^{R_k} \ldots \rightarrow^{R_{k+l-1}} N_{k+l}$ *mit* $\neg\Pi(N_j, R_j)$ $(k \leq j < k + l)$ *und* R *ist äquivalent zu* $R_\Pi \times R_{\neg\Pi}$.

Beispiele

Beispiel 11.2

- $\Pi(M, R)$ *gdw* R *ist Redex in* M*. I und II gelten.*
- $\Pi(M, R)$ *gdw* R *ist Outermostredex in* M*. dann gelten Eigenschaften I und II: Zu I*



R_0, R_2, R_3 Outermostredexe
Sei S_i der Redex in $M_0 \to M_i$
Ang. nicht OM $\rightsquigarrow$ In M_1 wird durch die Reduktion von S_1 ein Redex (P) erzeugt, der R_1 überdeckt.

In $M_1 \to\!> M_3$ wird R_1 wieder outermost. D.h. P wird reduziert: In $M_1 \to\!> M_3$ werden jedoch nur Reste von S_2 reduziert und P ist nicht Rest da neu eingeführt.↯. II ist klar.

- $\Pi(M, R)$ *gdw* R *ist left-most Redex in* M*. I gilt. II nicht immer:* $F(x, b) \to d, a \to b, c \to c :: F(c, a) \to F(c, b)$