Vorlesung

Termersetzungssysteme

Prof. Jürgen Giesl

SS 2002

Inhaltsverzeichnis

1 Einleitung **5**

2 Grundlagen **7**
2.1 Syntax von Gleichungssystemen 7
2.2 Semantik von Gleichungssystemen 8

3 Termersetzung und Deduktion **11**
3.1 Deduktion von Gleichungen 11
3.2 Kongruenzabschluss 19
3.3 Termersetzungssysteme 25

4 Terminierung **35**
4.1 NOETHERsche Induktion 35
4.2 Entscheidbarkeitsresultate zur Terminierung 36
4.3 Terminierung mit Reduktionsrelation 39
4.4 Simplifikationsordung und rekursive Pfadordnung 41

5 Konfluenz **49**
5.1 Unifikation 49
5.2 Lokale Konfluenz und kritische Paare 53

6 Vervollständigung **59**
6.1 Das grundlegende Vervollständigungs-Verfahren 59
6.2 Ein verbesserter Vervollständigungsalgorithmus 62

Kapitel 1

Einleitung

Termersetzungssysteme dienen zum Rechnen und automatischen Beweisen mit Gleichungen und sind daher grundlegend in der Informatik. Einige Anwendungsgebiete von Termersetzungssystemen sind

- Spezifikationen von Programmen
- automatisierte Analyse und Verifikation von Programmen
- Ausführung von Programmen
- symbolisches Rechnen
- ...

Beispiel (Datenstruktur der natürlichen Zahlen):

Man stelle die natürlichen Zahlen durch Terme aus den Funktionssymbolen (Konstruktoren) $\mathcal{O}$ und *succ* dar.

- $\mathcal{O} \mathrel{\widehat{=}} 0$
- $succ(\mathcal{O}) \mathrel{\widehat{=}} 1$
- $succ\big(succ(\mathcal{O})\big) \mathrel{\widehat{=}} 2$
- ...

Beispiel (Additionsalgorithmus):

$$\begin{aligned} plus(\mathcal{O}, y) &\equiv y \\ plus\big(succ(x), y\big) &\equiv succ\big(plus(x, y)\big) \end{aligned}$$

Die obigen beiden Zeilen stellen bereits ein *funktionales Programm* dar.

Berechne "$2 + 1$":

$$\begin{aligned} & plus\Big(\overbrace{succ\big(succ(\mathcal{O})\big)}^{\mathrel{\widehat{=}}2}, \overbrace{succ(\mathcal{O})}^{\mathrel{\widehat{=}}1}\Big) \\ \to\ & succ\Big(plus\big(succ(\mathcal{O})\big), succ(\mathcal{O})\Big) \\ \to\ & succ\Big(succ\big(plus(\mathcal{O}, succ(\mathcal{O}))\big)\Big) \\ \to\ & succ\Big(succ\big(succ(\mathcal{O})\big)\Big) \mathrel{\widehat{=}} 3 \end{aligned}$$

Da hier die Anwendung von "$\equiv$" *von links nach rechts* erfolgt, verwendet man stattdessen auch "$\rightarrow$". Somit erhält man das *Termersetzungssystem*:

$$\begin{aligned} plus(\mathcal{O}, y) &\rightarrow y \\ plus\big(succ(x), y\big) &\rightarrow succ\big(plus(x, y)\big) \end{aligned}$$

wobei jede der beiden Zeilen *Regel* heißt.

Beispiel (Datentyp "Gruppe"):

$$\begin{aligned} f\big(x, f(y, z)\big) &\equiv f\big(f(x, y), z\big) \\ f(x, e) &\equiv x \\ f\big(x, i(x)\big) &\equiv e \end{aligned}$$

Hierbei ist f eine binäre Operation, e das neutrale Element der Gruppe und i die inverse Funktion.

Kapitel 2

Grundlagen

In diesem Kapitel soll die Sprache der Terme und der Gleichungen festgelegt werden.

2.1 Syntax von Gleichungssystemen

Wir definieren zunächst das "Alphabet" aus denen die Terme gebildet werden:

Definition 2.1 (Signatur) *Eine **Signatur** $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \ldots = \bigcup_{i \in \mathbb{N}} \Sigma_i$ ist eine Vereinigung von paarweise disjunkten endlichen Mengen Σ_i. Jedes $f \in \Sigma_i$ heißt i-stelliges **Funktionssymbol**. Die Elemente von Σ_0 heißen **Konstanten**. Σ_0 darf nicht leer, Σ muss endlich sein.*

Beispiel:

$\Sigma_0 = \{\mathcal{O}\}$, $\Sigma_1 = \{succ\}$, $\Sigma_2 = \{plus\}$, $\Sigma_i = \emptyset$ $(i \geq 3)$

Definition 2.2 (Term) *Sei Σ eine Signatur und $\mathcal{V}$ eine nicht-leere (abzählbar) unendliche Menge mit $\mathcal{V} \cap \Sigma = \emptyset$. Elemente von $\mathcal{V}$ heißen **Variablen**. $\mathcal{T}(\Sigma, \mathcal{V})$ bezeichnet die Menge der **Terme** über Σ und $\mathcal{V}$. $\mathcal{T}(\Sigma, \mathcal{V})$ ist die kleinste Teilmenge von $\big(\Sigma \cup \mathcal{V} \cup \{(,), ,\}\big)^*$ mit*

(1) $\mathcal{V} \subseteq \mathcal{T}(\Sigma, \mathcal{V})$

(2) $f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, \mathcal{V})$, *falls* $f \in \Sigma_n$, $n \geq 0$, $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{V})$

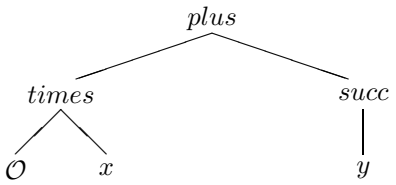
*$\mathcal{T}(\Sigma)$ steht für $\mathcal{T}(\Sigma, \emptyset)$, d. h. für die Menge der **Grundterme** (= Terme ohne Variablen). Für einen Term t bezeichnet $\mathcal{V}(t)$ die Menge der Variablen in t und für $T \subseteq \mathcal{T}(\Sigma, \mathcal{V})$ definieren wir $\mathcal{V}(T) := \bigcup_{t \in T} \mathcal{V}(t)$.*

Beispiel:

Σ wie oben, $\mathcal{V} := \{x, y, z\}$

x, y
$succ(x)$
$succ\big(succ(x)\big)$
$plus\big(x, succ(y)\big)$
$\mathcal{O}$
$succ(\mathcal{O})$
$\vdots$
sind Terme

$plus\big(times(\mathcal{O}, x), succ(y)\big)$



Terme $\widehat{=}$ Vielwegbäume

Definition 2.3 (Teilterm) *Ein Term q heißt **Teilterm** von einem Term t ($t \trianglerighteq q$) $:\curvearrowright\curvearrowright$ $t = q$ oder $t = f(t_1, \ldots, t_n)$ und $t_i \trianglerighteq q$ für ein $i \in \{1, \ldots, n\}$. $t_1, \ldots, t_n$ sind die **direkten Teilterme** von t. q ist **echter Teilterm** von t ($t \triangleright q$) $:\curvearrowright\curvearrowright$ $t \trianglerighteq q$ und $t \neq q$.*

Beispiel:

$plus\big(times(\mathcal{O}, x), succ(y)\big) \trianglerighteq times(\mathcal{O}, x)$
$plus\big(times(\mathcal{O}, x), succ(y)\big) \trianglerighteq \mathcal{O}$

Definition 2.4 (Gleichung) *Für $t_1, t_2 \in \mathcal{T}(\Sigma, \mathcal{V})$ heißt $t_1 \equiv t_2$ **(Term-)Gleichung**. Die Menge $\mathcal{E}$ von Gleichungen heißt **Gleichungssystem**.*

Beachte:

- "$\equiv$" ist ein rein *syntaktisches* Symbol, d. h. "$t_1 \equiv t_2$" muss keine "wahre" Aussage sein.
- Gleichung $\widehat{=}$ Paar von Termen (t_1, t_2)
- Gleichungssystem $\widehat{=}$ beliebige Teilmenge von $\mathcal{T}(\Sigma, \mathcal{V}) \times \mathcal{T}(\Sigma, \mathcal{V})$

2.2 Semantik von Gleichungssystemen

Gleichungen entsprechen Axiomen, die Programme bzw. Datenstrukturen festlegen. Es stellen sich die Fragen:

- Welche Aussagen gelten für diese Programme/Datenstrukturen?
- Welche Aussagen folgen aus den Axiomen?

Zur Beantwortung dieser Fragen muss zunächst die Semantik von Termen und Gleichungen geklärt werden. Dazu führen wir die Begriffe *Interpretation* und *Algebra* ein.

Definition 2.5 (Interpretation, Algebra) *Für eine Signatur Σ ist die **Σ-Interpretation** $\mathcal{I}$ ein Tripel $\mathcal{I} = \langle A, \alpha, \beta \rangle$, wobei*

- *A: **Träger** der Interpretation (beliebige Menge mit $A \neq \emptyset$)*
- *α ordnet jedem Funktionssymbol $f \in \Sigma$ eine Funktion $\alpha_f : \underbrace{\mathcal{A} \times \ldots \times \mathcal{A}}_{n\text{-mal}} \longrightarrow \mathcal{A}$ zu*
- *$\beta : \mathcal{V} \longrightarrow A$ heißt **Variablenbelegung***

Zu jeder Interpretation $\mathcal{I}$ erhält man eine Funktion $\mathcal{I} : \mathcal{T}(\Sigma, \mathcal{V}) \longrightarrow A$ wie folgt:

- *$\mathcal{I}(x) := \beta(x)$ für alle $x \in \mathcal{V}$*
- *$\mathcal{I}\big(f(t_1, \ldots, t_n)\big) := \alpha_f\big(\mathcal{I}(t_1), \ldots, \mathcal{I}(t_n)\big)$ $\forall f \in \Sigma, \forall t_i \in \mathcal{T}(\Sigma, \mathcal{V})$*

*$\mathcal{I}(t)$ heißt **Interpretation** des Terms t unter $\mathcal{I}$. Eine Σ-Interpretation ohne Variablenbelegung $\langle A, \alpha \rangle$ heißt **Σ-Algebra**.*

Beispiel:

$\mathcal{I} := \langle A, \alpha, \beta \rangle$

- $A = \mathbb{N} = \{0, 1, 2, \ldots\}$
- $\alpha_{\mathcal{O}} = 0$
- $\alpha_{succ}(n) = n + 1 \quad$ (für alle $n \in \mathbb{N}$)
- $\alpha_{plus}(n, m) = n + m$
- $\alpha_{times}(n, m) = n \cdot m$
- $\beta(x) = 5$
- $\beta(y) = 3$

$\mathcal{I}' := \langle A, \alpha, \beta' \rangle$

- $\beta'(x) = 2$
- $\beta'(y) = 1$

$\mathcal{I}'' := \langle A'', \alpha'', \beta \rangle$

- $A = \mathbb{Q}$
- $\alpha''_{\mathcal{O}} = 0$
- $\alpha''_{succ}(n) = n + 1 \quad$ (für alle $n \in \mathbb{Q}$)
- $\alpha''_{plus}(n, m) = \begin{cases} \frac{n}{m} & : \quad m \neq 0 \\ 0 & : \quad m = 0 \end{cases}$
- $\alpha''_{times}(n, m) = n \cdot m$

$$\mathcal{I}\big(plus(succ(x), y)\big) = \alpha_{plus}\big(\overbrace{\alpha_{succ}\big(\underbrace{\beta(x)}_{=5}\big)}^{=6}, \underbrace{\beta(y)}_{=3}\big) = 6 + 3 = 9$$

$$\mathcal{I}'\big(plus(succ(x), y)\big) = \alpha_{plus}\big(\overbrace{\alpha_{succ}\big(\underbrace{\beta'(x)}_{=2}\big)}^{=3}, \underbrace{\beta'(y)}_{=1}\big) = 3 + 1 = 4$$

$$\mathcal{I}''\big(plus(succ(x), y)\big) = \alpha''_{plus}\big(\overbrace{\alpha''_{succ}\big(\underbrace{\beta(x)}_{=5}\big)}^{=6}, \underbrace{\beta(y)}_{=3}\big) = \tfrac{6}{3} = 2$$

Definition 2.6 (Erfüllbarkeit, Modell) *Wir definieren*

- *Eine Interpretation $\mathcal{I} = \langle A, \alpha, \beta \rangle$ **erfüllt** eine Gleichung $t_1 \equiv t_2$ ($\mathcal{I} \models t_1 \equiv t_2$) $:\curvearrowright$ $\mathcal{I}(t_1) = \mathcal{I}(t_2)$.*
- *Eine Algebra $\mathcal{A} = \langle A, \alpha \rangle$ **erfüllt** eine Gleichung $t_1 \equiv t_2$ ($\mathcal{A} \models t_1 \equiv t_2$) $:\curvearrowright$ $\mathcal{I} \models t_1 \equiv t_2$ für alle Interpretationen der Form $\mathcal{I} = \langle A, \alpha, \beta \rangle$ (d. .h. für alle Variablenbelegungen β). Man sagt auch: "$\mathcal{A}$ ist **Modell** von $t_1 \equiv t_2$".*
- *Eine Algebra $\mathcal{A}$ ist **Modell** einer Gleichungsmenge $\mathcal{E}$ ($\mathcal{A} \models \mathcal{E}$) $:\curvearrowright$ $\mathcal{A} \models t_1 \equiv t_2$ für alle $t_1 \equiv t_2 \in \mathcal{E}$.*
- *Aus einer Gleichungsmenge $\mathcal{E}$ **folgt** die Gleichung $t_1 \equiv t_2$ ($\mathcal{E} \models t_1 \equiv t_2$) $:\curvearrowright$ für alle Algebren $\mathcal{A}$ mit $\mathcal{A} \models \mathcal{E}$ gilt: $\mathcal{A} \models t_1 \equiv t_2$. Anstelle von $\emptyset \models t_1 \equiv t_2$ schreibt man auch $\models t_1 \equiv t_2$. Solche Gleichungen heißen **allgemeingültig**.*

- *Wir definieren die Relation* $\equiv_{\mathcal{E}} \subseteq \mathcal{T}(\Sigma, \mathcal{V}) \times \mathcal{T}(\Sigma, \mathcal{V})$ *als* $s \equiv_{\mathcal{E}} t :\curvearrowright\curvearrowright \mathcal{E} \models s \equiv t$.

Beispiel:

Seien $\mathcal{I}$, $\mathcal{I}'$ und $\mathcal{I}''$ wie oben

- $\mathcal{I} \models plus(x, y) \equiv succ(succ(succ(succ(succ(succ(succ(succ(\mathcal{O}))))))))$
- $\mathcal{I}' \not\models plus(x, y) \equiv succ(succ(succ(succ(succ(succ(succ(succ(\mathcal{O}))))))))$
- $\mathcal{I}'' \not\models plus(x, y) \equiv plus(y, x)$

Sei $\mathcal{A} := (A, \alpha)$

- $\mathcal{A} \not\models plus(x, y) \equiv succ(succ(succ(succ(succ(succ(succ(succ(\mathcal{O}))))))))$
- $\mathcal{A} \models plus(x, y) \equiv plus(y, x)$
- $\{plus(x, y) \equiv plus(y, x)\} \models plus\big(\mathcal{O}, succ(\mathcal{O})\big) \equiv plus\big(succ(\mathcal{O}), \mathcal{O}\big)$
- $\emptyset \models \mathcal{O} \equiv \mathcal{O}$

Wortproblem:

Gilt $s \equiv_{\mathcal{E}} t$ für Terme s, t und Gleichungssystem $\mathcal{E}$?

Beispiel:

$\mathcal{E} := \{plus(x, y) \equiv plus(y, x)\}$

- $plus\big(\mathcal{O}, succ(\mathcal{O})\big) \equiv_{\mathcal{E}} plus\big(succ(\mathcal{O}), \mathcal{O}\big)$?
- $i\big(i(x)\big) \equiv_{\mathcal{E}} x$?

Kapitel 3

Termersetzung und Deduktion von Gleichungen

Um das *Wortproblem* automatisch (z. B. rechnergestützt) zu lösen, werden wir in diesem Kapitel drei Verfahren kennenlernen:

- naives Verfahren
- Verfahren, wenn $\mathcal{E}$ keine Variablen enthält: *Kongruenzabschluss*
- Verfahren für "beliebiges" $\mathcal{E}$: *Termersetzungssysteme*

3.1 Deduktion von Gleichungen

Das vorliegende Ziel ist es nun zu untersuchen, ob $s \equiv_{\mathcal{E}} t$ gilt, d. h. ob die *Gleichung* $s \equiv t$ aus dem *Gleichungssystem* $\mathcal{E}$ folgt. Da die semantische Definition unbrauchbar ist, um dies systematisch oder automatisch zu überprüfen, stellen wir der Semantik einen *syntaktischen Kalkül* gegenüber, der automatisch abgeleitet werden kann.

Definition 3.1 (Substitution, Matching) *Wir definieren*

- *Eine Abbildung* $\sigma : \mathcal{V} \longrightarrow \mathcal{T}(\Sigma, \mathcal{V})$ *heißt* ***Subsitution*** $:\Leftrightarrow \sigma(x) \neq x$ *nur für endlich viele* $x \in \mathcal{V}$.
- $SUB(\Sigma, \mathcal{V})$ *ist die* ***Menge aller Substitutionen*** *über* Σ *und* $\mathcal{V}$.
- $id \in SUB(\Sigma, \mathcal{V})$ *ist die* ***identische Substitution*** *mit* $id(x) = x \ \forall x \in \mathcal{V}$.
- $DOM(\sigma) := \{x \in \mathcal{V} \mid \sigma(x) \neq x\}$ *heißt die* ***Domain*** *von* σ. *Da die Domain endlich ist, kann man* σ *darstellen als* $\{x/\sigma(x) \mid x \in DOM(\sigma)\}$.
- σ *heißt* ***Grundsubstitution*** $:\Leftrightarrow \sigma(x) \in \mathcal{T}(\Sigma) \ \forall x \in DOM(\sigma)$.
- *Substitutionen werden zu der Abbildung* $\sigma : \mathcal{T}(\Sigma, \mathcal{V}) \longrightarrow \mathcal{T}(\Sigma, \mathcal{V})$ *erweitert:*

$$\sigma\big(f(t_1, \ldots, t_n)\big) \quad := \quad f\big(\sigma(t_1), \ldots, \sigma(t_n)\big)$$

- *Für zwei Terme* s, t *sagen wir:* s ***matcht*** $t :\Leftrightarrow$ *es existiert eine Substitution* σ *mit* $\sigma(s) = t$. σ *heißt dann* ***Matcher*** *von* s *und* t.

Beispiel:

$\mathcal{V} := \{x, y, z, u\}$
$\sigma := \{x/plus(x,y), y/\mathcal{O}, z/succ(z)\}$

- $\sigma(x) = plus(x,y)$
- $\sigma(u) = u$
- $DOM(\sigma) = \{x, y, z\}$
- $\sigma\big(plus(x,y)\big) = plus\big(\sigma(x), \sigma(y)\big) = plus\big(plus(x,y), \mathcal{O}\big)$

Der Term $plus(x,y)$ *matcht* den Term $plus\big(plus(x,y), \mathcal{O}\big)$ mit *Matcher* σ. Der Term $plus(\mathcal{O}, y)$ *matcht* den Term $plus\big(\mathcal{O}, succ(\ldots(succ(\mathcal{O}))\ldots)\big)$ mit *Matcher* $\{y/succ(\ldots(succ(\mathcal{O}))\ldots)\}$.

Schreibweisen:

- x^* steht für Tupel von Variablen $x_1, \ldots, x_n$
- t^* steht für Tupel von Termen $t_1, \ldots, t_n$
- $\{x^*/t^*\}$ steht für $\{x_1/t_1, \ldots, x_n/t_n\}$
- $t\sigma$ steht für $\sigma(t)$

Definition 3.2 (Stabilität) *Eine Relation $\rightarrow \subseteq \mathcal{T}(\Sigma,\mathcal{V}) \times \mathcal{T}(\Sigma,\mathcal{V})$ heißt **stabil** oder **abgeschlossen unter Substitution** :$\curvearrowright\curvearrowright$ für alle Terme t_1, t_2 und für alle Substitutionen σ gilt: aus $t_1 \rightarrow t_2$ folgt $t_1\sigma \rightarrow t_2\sigma$.*

Beweistechnik:
Strukturelle Induktion über Terme

Ziel: Zeige, dass die Aussage $\varphi(t)$ für alle Terme $t \in \mathcal{T}(\Sigma,\mathcal{V})$ gilt.

Vorgehen:

(1) I.A.: Zeige $\varphi(x)$ für alle $x \in \mathcal{V}$

(2) I.V.: Es gelte $\varphi(t_1), \ldots, \varphi(t_n)$

(3) I.S.: Zeige $\varphi\big(f(t_1, \ldots, t_n)\big)$ für alle $f \in \Sigma$, $t_i \in \mathcal{T}(\Sigma,\mathcal{V})$

Unser Ziel ist nun die syntaktische Analyse von $\equiv_{\mathcal{E}}$. Wir gehen hierzu in zwei Schritten vor:

(1) *Stabilität* von $\equiv_{\mathcal{E}}$
Was passiert mit Deutungen von Termen, wenn man Substitution anwendet?

(2) *Monotonie* von $\equiv_{\mathcal{E}}$
Was passiert mit Deutungen von Termen, wenn man sie in einen Kontext legt?

Lemma 3.1 (Stabilität von $\equiv_{\mathcal{E}}$) *Sei $\mathcal{I} := (A, \alpha, \beta)$ eine Σ-Interpretation, $s, t \in \mathcal{T}(\Sigma,\mathcal{V})$, $\sigma \in SUB(\Sigma,\mathcal{V})$. Sei $\mathcal{I}' := (A, \alpha, \beta')$ mit $\beta'(x) := \mathcal{I}(x\sigma)$ für alle $x \in \mathcal{V}$. Dann gilt:*

(1) $\mathcal{I}(t\sigma) = \mathcal{I}'(t)$ (Substitutionslemma)

(2) $\mathcal{I} \models s\sigma \equiv t\sigma \curvearrowright\!\!\curvearrowright \mathcal{I}' \models s \equiv t$

(3) Für alle Σ*-Algebren* $\mathcal{A}$ *gilt:* $\mathcal{A} \models s \equiv t \curvearrowright \mathcal{A} \models s\sigma \equiv t\sigma$

(4) Für alle Gleichungssysteme $\mathcal{E}$ *gilt:* $s \equiv_{\mathcal{E}} t \curvearrowright s\sigma \equiv_{\mathcal{E}} t\sigma$

Beweis:

(1) durch *strukturelle Induktion* über t

I.A.: $t = x \in \mathcal{V}$
$\mathcal{I}'(x) = \beta'(x) = \mathcal{I}(x\sigma)$ ✓

I.S.: $t = f(t_1, \ldots, t_n)$

$$\begin{aligned}\mathcal{I}\big(f(t_1,\ldots,t_n)\sigma\big) &= \mathcal{I}\big(f(t_1\sigma,\ldots,t_n\sigma)\big)\\ &= \alpha_f\big(\underbrace{\mathcal{I}(t_1\sigma)}_{\mathcal{I}'(t_1)},\ldots,\underbrace{\mathcal{I}(t_n\sigma)}_{\mathcal{I}'(t_n)}\big) \quad \text{(I.V.)}\\ &= \alpha_f\big(\mathcal{I}'(t_1),\ldots,\mathcal{I}'(t_n)\big)\\ &= \mathcal{I}'\big(f(t_1,\ldots,t_n)\big)\ \checkmark\end{aligned}$$

(2) $\mathcal{I} \models s\sigma \equiv t\sigma \curvearrowright\!\!\curvearrowright \underbrace{\mathcal{I}(s\sigma)}_{\mathcal{I}'(s)} = \underbrace{\mathcal{I}(t\sigma)}_{\mathcal{I}'(t)}$
$\curvearrowright\!\!\curvearrowright \mathcal{I}' \models s \equiv t$

(3) Sei $\mathcal{A} := \langle A, \alpha\rangle$ mit $\mathcal{A} \models s \equiv t$
$\mathcal{A} \models s \equiv t \curvearrowright \mathcal{I}' \models s \equiv t$ mit $\mathcal{I}' := \langle A, \alpha, \beta'\rangle$ und $\beta'(x) = \mathcal{I}(x\sigma)$ für alle $x \in \mathcal{V} \overset{(2)}{\curvearrowright\!\!\curvearrowright} \mathcal{I} \models s\sigma \equiv t\sigma$

(4) $s \equiv_{\mathcal{E}} t \curvearrowright\!\!\curvearrowright \mathcal{E} \models s \equiv t \curvearrowright\!\!\curvearrowright$ für alle Algebren $\mathcal{A}$ mit $\mathcal{A} \models \mathcal{E}$ gilt: $\mathcal{A} \models s \equiv t$
$\overset{(3)}{\curvearrowright}$ für alle Algebren $\mathcal{A}$ mit $\mathcal{A} \models \mathcal{E}$ gilt: $\mathcal{A} \models s\sigma \equiv t\sigma \curvearrowright\!\!\curvearrowright s\sigma \equiv_{\mathcal{E}} t\sigma$

q.e.d.

Beispiel:

$\mathcal{I} := (A, \alpha, \beta)$ mit
$A := \mathbb{N}$, $\alpha_{\mathcal{O}} := 0$, $\alpha_{succ}(n) := n + 1$, $\alpha_{plus}(n, m) := n + m$, $\beta(x) := 5$
$\sigma := \{x/succ(x)\}$

$\mathcal{I}' := (A, \alpha, \beta')$ mit
$\beta'(x) = \mathcal{I}(x\sigma) = \mathcal{I}\big(succ(x)\big) = \alpha_{succ}\big(\underbrace{\beta(x)}_{=5}\big) = 6$

$t := plus(x, x)$
$\mathcal{I}\big(\underbrace{plus(x,x)\sigma}_{=plus\left(succ(x),succ(x)\right)}\big) = 12 = \mathcal{I}'\big(plus(x,x)\big)$

Definition 3.3 (Stelle) *Für einen Term* t *ist* $Occ(t)$ *("occurences") die Menge seiner* ***Stellen*** *(oder "Positionen") die kleinste Teilmenge von* $\mathbb{N}^*$ *mit:*

(1) $\varepsilon \in Occ(t)$

(2) $i\pi \in Occ(t)$*, falls* $t = f(t_1, \ldots, t_n)$ *und* $\pi \in Occ(t_i)$

Für t *mit* $\pi \in Occ(t)$ *bezeichnet* $t|_\pi$ *den Teilterm von* t *an Stelle* π*, wobei*

(1) $t|_\varepsilon = t$

(2) $f(t_1, \ldots, t_n)|_{i\pi} = t_i|_\pi$

Für t mit $\pi \in Occ(t)$ und Term r bezeichnet $t[r]_\pi$ den Term, der aus t durch Ersetzen von $t|_\pi$ durch r entsteht, d. h.:

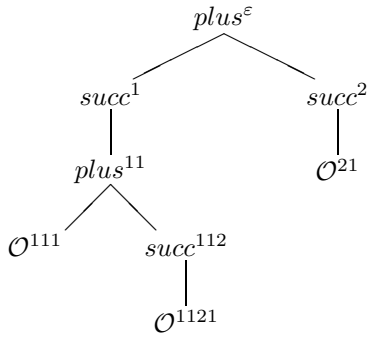
(1) $t[r]_\varepsilon = r$

(2) $f(t_1, \ldots, t_n)[r]_{i\pi} = f(t_1, \ldots, t_i[r]_\pi, \ldots, t_n)$

Für zwei Stellen π_1, π_2 sagen wir

- *π_1 ist **unterhalb** von π_2 ($\pi_1 >_{\mathbb{N}^*} \pi_2$)*
  $:\curvearrowright$ π_2 ist echtes Anfangsstück von π_1 ($\pi_1 = \pi_2\pi$, $\pi \in \mathbb{N}^+$)
- *π_1 und π_2 sind voneinander **unabhängig** ($\pi_1 \perp \pi_2$)*
  $:\curvearrowright$ $\pi_1 \not>_{\mathbb{N}^*} \pi_2$, $\pi_2 \not>_{\mathbb{N}^*} \pi_1$ und $\pi_1 \neq \pi_2$

Beispiel:



$t = plus\Big(succ\big(plus(\mathcal{O}, succ(\mathcal{O})), succ(\mathcal{O})\big)\Big)$

- $t|_2 = succ(\mathcal{O})$
- $t|_{21} = \mathcal{O}$
- $t[times(\mathcal{O}, \mathcal{O})]_{112} = plus\Big(succ\big(plus(\mathcal{O}, times(\mathcal{O}, \mathcal{O}))\big), succ(\mathcal{O})\Big)$
- $21 >_{\mathbb{N}^*} 2 >_{\mathbb{N}^*} \varepsilon$
- $21 \perp 11$

Definition 3.4 (Monotonie) *Eine Relation $\rightarrow \subseteq \mathcal{T}(\Sigma, \mathcal{V}) \times \mathcal{T}(\Sigma, \mathcal{V})$ heißt **monoton** $:\curvearrowright$ für alle Terme t_1, t_2, q und alle $\pi \in Occ(q)$ gilt: aus $t_1 \rightarrow t_2$ folgt, dass auch $q[t_1]_\pi \rightarrow q[t_2]_\pi$ gilt.*

Beweistechnik:
strukturelle Induktion über Stellen

Ziel: Zeige, dass die Aussage $\varphi(\pi)$ für alle $\pi \in \mathbb{N}^*$ gilt.

Vorgehen:

(1) I.A.: Zeige $\varphi(\varepsilon)$

(2) I.V.: Es gelte $\varphi(\pi')$ für $\pi' \in \mathbb{N}^*$

(3) I.S.: Zeige $\varphi(i\pi')$ (oder $\varphi(\pi' i)$) für alle $i \in \mathbb{N}$, $\pi' \in \mathbb{N}^*$

Lemma 3.2 (Monotonie von $\equiv_\mathcal{E}$) *Seien $s, t, q \in \mathcal{T}(\Sigma, \mathcal{V})$ und $\pi \in Occ(q)$. Dann gilt:*

(1) Für alle Algebren $\mathcal{A}$: $\mathcal{A} \models s \equiv t \curvearrowright \mathcal{A} \models q[s]_\pi \equiv q[t]_\pi$

(2) Für alle Gleichungssyteme $\mathcal{E}$: $s \equiv_\mathcal{E} t \curvearrowright q[s]_\pi \equiv_\mathcal{E} q[t]_\pi$

Beweis:

(1) durch *strukturelle Induktion* über π

I.A.: $\mathcal{A} \models s \equiv t \curvearrowright \mathcal{A} \models \underbrace{q[s]_\varepsilon}_{=s} \equiv_\mathcal{E} \underbrace{q[t]_\varepsilon}_{=t}$ ✓

I.S.: Wenn $i\pi' \in Occ(q)$, dann hat q die Gestalt $f(q_1, \ldots, q_n)$ und $\pi' \in Occ(q_i)$
$\curvearrowright q[s]_{i\pi'} = f(q_1, \ldots, q_i[s]_{\pi'}, \ldots, q_n)$
$\curvearrowright q[t]_{i\pi'} = f(q_1, \ldots, q_i[t]_{\pi'}, \ldots, q_n)$
Sei $\mathcal{A} = (A, \alpha)$ mit $\mathcal{A} \models s \equiv t$
z. z: $\forall \mathcal{I} : \mathcal{I} \models f(q_1, \ldots, q_i[s]_{\pi'}, \ldots, q_n) \equiv f(q_1, \ldots, q_i[t]_{\pi'}, \ldots, q_n)$, also:
$\alpha_f(\mathcal{I}(q_1), \ldots, \underbrace{\mathcal{I}(q_i[s]_{\pi'})}_{\text{(i)}}, \ldots, \mathcal{I}(q_n)) \equiv \alpha_f(\mathcal{I}(q_1), \ldots, \underbrace{\mathcal{I}(q_i[t]_{\pi'})}_{\text{(ii)}}, \ldots, \mathcal{I}(q_n))$
Nach I.V. gilt: $\mathcal{A} \models s \equiv t \curvearrowright \mathcal{A} \models q_i[s]_{\pi'} \equiv q_i[t]_{\pi'}$
$\curvearrowright \mathcal{I} \models q_i[s]_{\pi'} \equiv q_i[t]_{\pi'} \curvearrowright \underbrace{\mathcal{I}(q_i[s]_{\pi'})}_{\text{(i)}} = \underbrace{\mathcal{I}(q_i[t]_{\pi'})}_{\text{(ii)}}$

(2) $s \equiv_\mathcal{E} t \curvearrowleft\curvearrowright \mathcal{E} \models s \equiv t$
$\curvearrowleft\curvearrowright$ für alle $\mathcal{A}$ mit $\mathcal{A} \models \mathcal{E}$ gilt: $\mathcal{A} \models s \equiv t$
$\curvearrowright$ für alle $\mathcal{A}$ mit $\mathcal{A} \models \mathcal{E}$ gilt: $\mathcal{A} \models q[s]_\pi \equiv q[t]_\pi$
$\curvearrowleft\curvearrowright q[s]_\pi \equiv_\mathcal{E} q[t]_\pi$

q.e.d.

Definition 3.5 (Äquivalenzrelation) *Sei M eine Menge und $\rightarrow$ eine Relation über M, d. h. $\rightarrow \subseteq M \times M$. Dann heißt $\rightarrow$*

- ***reflexiv**, falls $t \rightarrow t$ für alle $t \in M$,*
- ***symmetrisch**, falls $t_1 \rightarrow t_2 \curvearrowright t_2 \rightarrow t_1$ für alle $t_1, t_2 \in M$,*
- ***transitiv**, falls $t_1 \rightarrow t_2 \wedge t_2 \rightarrow t_3 \curvearrowright t_1 \rightarrow t_3$ für alle $t_1, t_2, t_3 \in M$ und*
- ***Äquivalenzrelation**, falls $\rightarrow$ reflexiv, symmetrisch und transitiv ist.*

Beispiel:

- Die Relation $= \subseteq \mathbb{N}^2$ ist eine *Äquivalenzrelation.*
- Die Relation $\trianglerighteq \subseteq \mathcal{T}(\Sigma, \mathcal{V}) \times \mathcal{T}(\Sigma, \mathcal{V})$ ist *reflexiv* und *transitiv.*

Definition 3.6 (Reflexive, transitive, symmetrische Hülle) *Sei M eine Menge und $\rightarrow \subseteq M \times M$.*

- *Die **reflexive Hülle** von $\rightarrow$ ($\rightarrow^=$) ist die kleinste reflexive Relation, die $\rightarrow$ enthält, d. h.*
  - *$t_1 \rightarrow t_2 \curvearrowright t_1 \rightarrow^= t_2$ und*
  - *$t_1 \rightarrow^= t_1$.*
- *Die **transitive Hülle** von $\rightarrow$ ($\rightarrow^+$) ist die kleinste transitive Relation, die $\rightarrow$ enthält, d. h.*
  - *$t_1 \rightarrow t_2 \curvearrowright t_1 \rightarrow^+ t_2$ und*
  - *$t_1 \rightarrow t_2 \rightarrow^+ t_3 \curvearrowright t_1 \rightarrow^+ t_3$.*

- *Die **transitiv-reflexive Hülle** von $\rightarrow$ ($\rightarrow^*$) ist die kleinste transitive und reflexive Relation, die $\rightarrow$ enthält, d. h.*
  - $t_1 \rightarrow^+ t_2 \curvearrowright t_1 \rightarrow^* t_2$ *und*
  - $t_1 \rightarrow^* t_1$.
- *Die **symmetrische Hülle** von $\rightarrow$ ($\leftrightarrow$) ist die kleinste symmetrische Relation, die $\rightarrow$ enthält, d. h.*
  - $t_1 \leftrightarrow t_2 \curvearrowleft\curvearrowright t_1 \rightarrow t_2$ *oder* $t_2 \rightarrow t_1$.
- *Die **transitiv-reflexiv-symmetrische Hülle** von $\rightarrow$ ($\leftrightarrow^*$) ist die kleinste Äquivalenzrelation, die $\rightarrow$ enthält.*

Beispiel:

Sei $\succ \subseteq \mathbb{N}^2$ definiert durch $m \succ n :\curvearrowright m = n + 1$.

- $m \succ^= n \curvearrowright m = n + 1$ oder $m = n$
- $m \succ^+ n \curvearrowright m > n$
- $m \succ^* n \curvearrowright m \geq n$
- $m \prec\succ n \curvearrowright |m - n| = 1$
- $m \prec\succ^* n \curvearrowright m, n \in \mathbb{N}$

Definition 3.7 (Kongruenzrelation) *Eine Äquivalenzrelation $\rightarrow$ heißt **Kongruenzrelation** $:\curvearrowright$ für alle $f \in \Sigma$ und alle $s_1, \ldots, s_n, t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{V})$ gilt: wenn $s_1 \rightarrow t_1, \ldots, s_n \rightarrow t_n$, dann auch $f(s_1, \ldots, s_n) \rightarrow f(t_1, \ldots, t_n)$.*

Lemma 3.3 *Für alle Gleichungssysteme $\mathcal{E}$ ist $\equiv_\mathcal{E}$ eine Kongruenzrelation*

Beweis:

[Übung]

q.e.d.

Definition 3.8 (Ersetzungsrelation, Beweisrelation, Herleitung) *Für ein Gleichungssystem $\mathcal{E}$ ist die **Ersetzungsrelation** $\rightarrow_\mathcal{E} \subseteq \mathcal{T}(\Sigma, \mathcal{V}) \times \mathcal{T}(\Sigma, \mathcal{V})$ definiert durch*

$$s \rightarrow_\mathcal{E} t \curvearrowright s|_\pi = t_1\sigma \wedge t = s[t_2\sigma]_\pi$$

*für eine Stelle $\pi \in Occ(s)$, eine Gleichung $t_1 \equiv t_2 \in \mathcal{E}$ und eine Substitution $\sigma \in SUB(\Sigma, \mathcal{V})$. Die Relation $\leftrightarrow^*_\mathcal{E}$ heißt **Beweisrelation** von $\mathcal{E}$. Wir sagen $s \equiv t$ ist aus $\mathcal{E}$ **herleitbar** ($\mathcal{E} \vdash s \equiv t$), falls $s \leftrightarrow^*_\mathcal{E} t$.*

Beispiel:

$$\mathcal{E} = \{ \underbrace{plus(\mathcal{O}, y) \equiv y}_{(1)}, \underbrace{plus\big(succ(x), y\big) \equiv succ\big(plus(x, y)\big)}_{(2)} \}$$

Frage: Gilt $plus(succ(succ(\mathcal{O})), x) \equiv_\mathcal{E} plus(succ(\mathcal{O}), succ(x))$?

Wir zeigen stattdessen $plus(succ(succ(\mathcal{O})), x) \leftrightarrow^*_\mathcal{E} plus(succ(\mathcal{O}), succ(x))$:

$$
\begin{array}{lll}
\underline{plus(succ(succ(\mathcal{O})), x)} & \overset{(2)}{\rightarrow_{\mathcal{E}}} & succ(\underline{plus(succ(\mathcal{O}), x)}) \quad \pi = \varepsilon, \\
& & \qquad \sigma = \{x/succ(\mathcal{O}), y/x\} \\
& \overset{(2)}{\rightarrow_{\mathcal{E}}} & succ(succ(\underline{plus(\mathcal{O}, x)})) \quad \pi = 1, \\
& & \qquad \sigma = \{x/\mathcal{O}, y/x\} \\
& \overset{(1)}{\rightarrow_{\mathcal{E}}} & succ(\underline{succ(x)}) \quad \pi = 11, \\
& & \qquad \sigma = \{y/x\} \\
& \overset{(1)}{\leftarrow_{\mathcal{E}}} & \underline{succ(plus(\mathcal{O}, succ(x)))} \quad \pi = 1, \\
& & \qquad \sigma = \{y/succ(x)\} \\
& \overset{(2)}{\leftarrow_{\mathcal{E}}} & plus(succ(\mathcal{O}), succ(x)) \quad \pi = \varepsilon, \\
& & \qquad \sigma = \{x/\mathcal{O}, y/succ(x)\}
\end{array}
$$

Die Relation $\leftrightarrow_{\mathcal{E}}^*$ ist auf rein *syntaktische* Art und Weise definiert. Um $\leftrightarrow_{\mathcal{E}}^*$ für den Beweis von Wortproblemen benutzen zu können, muss der Zusammenhang zwischen $\equiv_{\mathcal{E}}$ und $\leftrightarrow_{\mathcal{E}}^*$ untersucht werden.

Lemma 3.4 *Sei $\mathcal{E}$ ein Gleichungssystem, $s, t, q \in \mathcal{T}(\Sigma, \mathcal{V})$, $\pi \in Occ(q)$, $\sigma \in SUB(\Sigma, \mathcal{V})$. Dann ist $\leftrightarrow_{\mathcal{E}}^*$ eine Kongruenzrelation und es gilt:*

(1) $s \rightarrow_{\mathcal{E}} t \curvearrowright s\sigma \rightarrow_{\mathcal{E}} t\sigma$ *und* $s \leftrightarrow_{\mathcal{E}}^* t \curvearrowright s\sigma \leftrightarrow_{\mathcal{E}}^* t\sigma$

(2) $s \rightarrow_{\mathcal{E}} t \curvearrowright q[s]_\pi \rightarrow_{\mathcal{E}} q[t]_\pi$ *und* $s \leftrightarrow_{\mathcal{E}}^* t \curvearrowright q[s]_\pi \leftrightarrow_{\mathcal{E}}^* q[t]_\pi$

Beweis:

[Übung]

q.e.d.

Satz 3.1 (Birkhoff, 1935) *Sei $\mathcal{E}$ ein Gleichungssystem. Dann sind die Relationen $\equiv_{\mathcal{E}}$ und $\leftrightarrow_{\mathcal{E}}^*$ identisch, d. h. $s \equiv_{\mathcal{E}} t \curvearrowleft\curvearrowright s \leftrightarrow_{\mathcal{E}}^* t$ für alle $s, t \in \mathcal{T}(\Sigma, \mathcal{V})$.*

Beweis:

Korrektheit ($s \leftrightarrow_{\mathcal{E}}^* t \curvearrowright s \equiv_{\mathcal{E}} t$):

Zeige zuerst: $s \leftrightarrow_{\mathcal{E}} t \curvearrowright s \equiv_{\mathcal{E}} t$
$s \leftrightarrow_{\mathcal{E}} t \curvearrowright$ es existiert $t_1 \equiv t_2$ oder $t_2 \equiv t_1$ in $\mathcal{E}$, so dass $s|_\pi = t_1\sigma$ und $t = s[t_2\sigma]_\pi$. Da $\equiv_{\mathcal{E}}$ symmetrisch ist, folgt $t_1 \equiv_{\mathcal{E}} t_2$. Aus der Stabilität von $\equiv_{\mathcal{E}}$ folgt $t_1\sigma \equiv_{\mathcal{E}} t_2\sigma$. Und es gilt $\underbrace{s[t_1\sigma]_\pi}_{=s} \equiv_{\mathcal{E}} \underbrace{s[t_2\sigma]_\pi}_{=t}$ wegen der Monotonie von $\equiv_{\mathcal{E}}$. ✓

Zeige nun: $s \leftrightarrow_{\mathcal{E}}^n t \curvearrowright s \equiv_{\mathcal{E}} t$ für alle $n \in \mathbb{N}$

I.A.: $s \leftrightarrow_{\mathcal{E}}^0 t \curvearrowright s = t \curvearrowright s \equiv_{\mathcal{E}} t$ (wegen Reflexivität von $\equiv_{\mathcal{E}}$)
I.S.: $s = s_0 \leftrightarrow_{\mathcal{E}} s_1 \leftrightarrow_{\mathcal{E}} \ldots \leftrightarrow_{\mathcal{E}} s_n = t$
$\curvearrowleft\curvearrowright \quad s \leftrightarrow_{\mathcal{E}}^{n-1} s_{n-1} \leftrightarrow_{\mathcal{E}} t$
$\overset{\text{I.V.}}{\curvearrowright} s \equiv_{\mathcal{E}} s_{n-1}$ und $s_{n-1} \equiv_{\mathcal{E}} t \curvearrowright s \equiv_{\mathcal{E}} t$ (wegen Transitivität von $\equiv_{\mathcal{E}}$)

Vollständigkeit ($s \equiv_{\mathcal{E}} t \curvearrowright s \leftrightarrow_{\mathcal{E}}^* t$):

$s \equiv_{\mathcal{E}} t$ bedeutet: jedes Modell $\mathcal{A}$ von $\mathcal{E}$ erfüllt auch $s \equiv t$. Daher ist das Ziel: definiere ein Modell $\mathcal{A} = \langle A, \alpha \rangle$ von $\mathcal{E}$ und eine Interpretation $\mathcal{I} = \langle A, \alpha, \beta \rangle$, so

dass $\mathcal{I} \models s \equiv t \curvearrowright\!\!\curvearrowright s \leftrightarrow_{\mathcal{E}}^* t$ für alle $s, t \in \mathcal{T}(\Sigma, \mathcal{V})$.
Zu jedem s bezeichnet $[s]_{\leftrightarrow_{\mathcal{E}}^*}$ die Äquivalenzklasse von s, d. h.

$$[s]_{\leftrightarrow_{\mathcal{E}}^*} = \{t \mid s \leftrightarrow_{\mathcal{E}}^* t\}$$

Bei jeder Äquivalenzrelation sind 2 Äquivalenzklassen entweder gleich oder disjunkt; daher lässt sich $\mathcal{T}(\Sigma, \mathcal{V})$ in disjunkte Äquivalenzklassen einteilen. Die Menge der Äquivalenzklassen heißt **Quotientenmenge**

$$\mathcal{T}(\Sigma, \mathcal{V})/_{\leftrightarrow_{\mathcal{E}}^*} = \{[s]_{\leftrightarrow_{\mathcal{E}}^*} \mid s \in \mathcal{T}(\Sigma, \mathcal{V})\}$$

Definiere $\mathcal{I}$ so, dass $\mathcal{I}(s) = [s]_{\leftrightarrow_{\mathcal{E}}^*}$, denn dann gilt:
$\mathcal{I} \models s \equiv t \curvearrowright\!\!\curvearrowright \mathcal{I}(s) = \mathcal{I}(t) \curvearrowright\!\!\curvearrowright [s]_{\leftrightarrow_{\mathcal{E}}^*} = [t]_{\leftrightarrow_{\mathcal{E}}^*} \curvearrowright\!\!\curvearrowright s \leftrightarrow_{\mathcal{E}}^* t$
Also: $\mathcal{I} := \langle A, \alpha, \beta \rangle$ mit

- $A := \mathcal{T}(\Sigma, \mathcal{V})/_{\leftrightarrow_{\mathcal{E}}^*}$
- $\alpha_f\big([t_1]_{\leftrightarrow_{\mathcal{E}}^*}, \ldots, [t_n]_{\leftrightarrow_{\mathcal{E}}^*}\big) := [f(t_1, \ldots, t_n)]_{\leftrightarrow_{\mathcal{E}}^*}$
- $\beta(x) := [x]_{\leftrightarrow_{\mathcal{E}}^*}$

Somit gilt: $\mathcal{I} \models s \equiv t \curvearrowright\!\!\curvearrowright s \leftrightarrow_{\mathcal{E}}^* t$

Noch zu zeigen: $\mathcal{A} := \langle A, \alpha \rangle \models \mathcal{E}$

Sei $u \equiv v \in \mathcal{E}$. Zu zeigen ist: $\mathcal{A} \models u \equiv v$, d. h. für alle $\mathcal{J} = \langle A, \alpha, \gamma \rangle$ gilt: $\mathcal{J} \models u \equiv v$, d. h. $\mathcal{J}(u) = \mathcal{J}(v)$.

Für jede Variable x ist $\gamma(x) = [Sx]_{\leftrightarrow_{\mathcal{E}}^*}$ für einen Term Sx. Sei σ die Substitution mit $\sigma(x) = Sx$ für alle Variablen x, die in $\mathcal{E}$ auftreten. Dann gilt $\mathcal{J}(t) = [t\sigma]_{\leftrightarrow_{\mathcal{E}}^*}$ für alle Terme t (strukturelle Induktion).

Für alle $u \equiv v \in \mathcal{E}$ gilt: $u \leftrightarrow_{\mathcal{E}}^* v \curvearrowright u\sigma \leftrightarrow_{\mathcal{E}}^* v\sigma \curvearrowright\!\!\curvearrowright \underbrace{[u\sigma]_{\leftrightarrow_{\mathcal{E}}^*}}_{\mathcal{J}(u)} = \underbrace{[v\sigma]_{\leftrightarrow_{\mathcal{E}}^*}}_{\mathcal{J}(v)}$

q.e.d.

Beispiel:

- $\mathcal{T}(\Sigma, \mathcal{V})/_{\leftrightarrow_{\mathcal{E}}^*} = \{[\mathcal{O}]_{\leftrightarrow_{\mathcal{E}}^*}, [succ(\mathcal{O})]_{\leftrightarrow_{\mathcal{E}}^*}, [x]_{\leftrightarrow_{\mathcal{E}}^*}, \ldots\}$
- $[\mathcal{O}]_{\leftrightarrow_{\mathcal{E}}^*} = \{\mathcal{O}, plus(\mathcal{O}, \mathcal{O}), plus(plus(\mathcal{O}, \mathcal{O}), \mathcal{O}), \ldots\}$
- $[succ(\mathcal{O})]_{\leftrightarrow_{\mathcal{E}}^*} = \{succ(\mathcal{O}), plus(\mathcal{O}, succ(\mathcal{O})), \ldots\}$
- $[x]_{\leftrightarrow_{\mathcal{E}}^*} = \{x, plus(\mathcal{O}, x), plus(x, \mathcal{O}), \ldots\}$
- $\ldots$

Beweisverfahren zur Überprüfung von $s \equiv_{\mathcal{E}} t$

(1) Erstelle einen Suchbaum mit s als Wurzel.

(2) Jeder Knoten mit Markierung s' hat als Kinder die Knoten mit den Markierungen u, wenn $s' \leftrightarrow_{\mathcal{E}} u$

(3) Wenn $s, u_1, \ldots, u_n$ eine Knotenmarkierung auf einem Pfad im Suchbaum von der Wurzel aus zu einem beliebigen Knoten u_n ist, so gilt: $s \leftrightarrow_{\mathcal{E}} u_1 \leftrightarrow_{\mathcal{E}} \ldots \leftrightarrow_{\mathcal{E}} u_n$, d. h. $s \leftrightarrow_{\mathcal{E}}^* u_n$.

(4) Überprüfe bei jedem Knoten, ob die Markierung $u_n = t$ ist. Falls $s \leftrightarrow_{\mathcal{E}}^* t$, so existert solch ein Knoten nach endlich vielen Schritten; andernfalls termiert das Verfahren nicht.

Dieses Verfahren ist ein *Semi-Entscheidungsverfahren* für das Wortproblem mit exponentiellem Aufwand.

Beachte:

Wenn $\mathcal{V}(t_1) = \mathcal{V}(t_2)$ für alle $t_1 \equiv t_2 \in \mathcal{E}$, dann hat jeder Knoten im Suchbaum maximal $2 \cdot |Occ(s)| \cdot |\mathcal{E}| < \infty$ viele Kinder (exponentielles Wachstum). Ohne die *Variablenbedingung* $\mathcal{V}(t_1) = \mathcal{V}(t_2)$ wird der Verzweigungsgrad im Suchbaum unendlich groß.

3.2 Der Kongruenzabschluss bei Grundidentitäten

Im Allgemeinen gibt es für das Wortproblem *kein* Entscheidungsverfahren, sondern lediglich ein *Semi-Entscheidungsverfahren*. Es wird aber entscheidbar, falls das Gleichungssystem und die zu untersuchenden Terme keine Variablen enthalten. Was auf den ersten Augenschein nicht brauchbar erscheint, ist aber für eine Reihe von Problemen dennoch interessant:

- *Compilerbau* (*common subexpression problem*):
  Untersuche, ob Teilausdrücke "gleich" sind, damit man nur einmal auswerten muss.
- *Programmverifikation*:
  Aus einem Verfahren für die Lösung des Wortproblems bei Grundidentitäten kann man ein Entscheidungsverfahren für die Allgemeingültigkeit universeller Formeln und für universelle Aussagen über rekursiv definierte Datenstrukturen gewinnen.
- *Kombination* von Entscheidungsverfahren

Definition 3.9 (Grundidentität) *Eine Termgleichung $s \equiv t$ heißt* ***Grundidentität*** $:\curvearrowright \mathcal{V}(s) = \mathcal{V}(t) = \emptyset$.

Beispiel:

Gegeben sei ein Programmfragment mit `i`, `j`, `k`, `l` und `m` Variablen für *Integer* und `f` und `g` für *Arrays von Integern*.

```
⋮
i = j;
k = l;
f[i] = g[k];
if(j == f[j]){
  m = g[l];
  ...              //Gilt hier f[m] == g[k]?
}
⋮
```

Die obige Frage bedeutet: gilt $f(m) \equiv_{\mathcal{E}} g(k)$?, wobei
$\mathcal{E} = \{i \equiv j, k \equiv l, f(i) \equiv g(k), j \equiv f(j), m \equiv g(l)\}$

Hierbei sind i, j, k, l, m Konstanten und f, g einstellige Funktionssymbole.

Da wir $f(m) \equiv g(k)$ *nicht für alle* Werte von m und k zeigen wollen, sondern nur für genau die Werte, die $\mathcal{E}$ "erfüllen", handelt es sich hierbei um das *Wortproblem bei Grundidentitäten.*

Im Folgenden wollen wir ein Verfahren entwickeln, dass das Wortproblem bei Grundidentitäten entscheidet. Da Grundidentitäten *keine Variablen* besitzen, ist die Stabilität von $\equiv_{\mathcal{E}}$ uninteressant. Die Idee des Verfahrens ist es daher, das zugrundeliegende Gleichungssystem $\mathcal{E}$ um die Gleichungen zu erweitern, die für die Reflexivität, Symmetrie und Transitivität sowie die Kongruenz gebraucht werden. Falls nun die Gleichung $s \equiv t$ in dieser Erweiterung vorkommt, gilt $s \equiv_{\mathcal{E}} t$. Wir werden sehen, dass dies leider auch nur ein *Semi-Entscheidungsverfahren* beschreibt.

Definition 3.10 *Sei $\mathcal{E}$ ein Gleichungssystem über Σ. Dann definieren wir*

- $R := \{t \equiv t \mid t \in \mathcal{T}(\Sigma)\}$
- $S(\mathcal{E}) := \{t \equiv s \mid s \equiv t \in \mathcal{E}\}$
- $T(\mathcal{E}) := \{s \equiv r \mid \exists t \in \mathcal{T}(\Sigma) : s \equiv t, t \equiv r \in \mathcal{E}\}$
- $C(\mathcal{E}) := \{f(s_1, \ldots, s_n) \equiv f(t_1, \ldots, t_n) \mid f \in \Sigma, s_1 \equiv t_1, \ldots, s_n \equiv t_n \in \mathcal{E}\}$

Beispiel:

- $R = \{i \equiv i, j \equiv j, f(i) \equiv f(i), \ldots\}$
- $S(\mathcal{E}) = \{j \equiv i, l \equiv k, g(k) \equiv f(i), \ldots\}$
- $T(\mathcal{E}) = \{i \equiv f(j)\}$
- $C(\mathcal{E}) = \{f(i) \equiv f(j), g(i) \equiv g(j), \ldots\}$

Am Beispiel sieht man, dass zwar alle Gleichungen in $R \cup S(\mathcal{E}) \cup T(\mathcal{E}) \cup C(\mathcal{E})$ aus $\mathcal{E}$ folgen, aber man erhält nicht alle aus $\mathcal{E}$ folgenden Gleichungen. Aus diesem Grund muss die Anwendnung von S, T und C wiederholt werden, und zwar unendlich oft. Das Resultat dieser wiederholten Anwendung nennt man *Kongruenzabschluss.*

Definition 3.11 (Kongruenzabschluss) *Für jede Menge von Grundidentitäten $\mathcal{E}$ definieren wir:*

(1) $\mathcal{E}_0 := \mathcal{E} \cup R$

(2) $\mathcal{E}_{i+1} := \mathcal{E}_i \cup S(\mathcal{E}_i) \cup T(\mathcal{E}_i) \cup C(\mathcal{E}_i)$

*Der **Kongruenzabschluss** (congruence closure) $CC(\mathcal{E})$ ist definiert durch*

$$CC(\mathcal{E}) := \bigcup_{i \in \mathbb{N}} \mathcal{E}_i$$

Beweisverfahren zur Überprüfung von $s \equiv_{\mathcal{E}} t$ bei Grundidentitäten

Berechne $\mathcal{E}_0, \mathcal{E}_1, \mathcal{E}_2, \ldots$ bis ein $\mathcal{E}_i$ erreicht wird, so dass $s \equiv t \in \mathcal{E}_i$. Dann gilt: $s \equiv_{\mathcal{E}} t$.

Hier haben wir es – wie oben bereits erwähnt – immer noch mit einem *Semi-Entscheidungsverfahren* zu tun, da im Allgemeinen stets gilt: $\mathcal{E}_i \subset \mathcal{E}_{i+1}$, so dass der Kongruenzabschluss *unendlich groß* wird.

Satz 3.2 *Sei $\mathcal{E}$ eine Menge von Grundidentitäten über Σ und $s,t \in \mathcal{T}(\Sigma)$. Dann gilt:*

$$s \equiv_{\mathcal{E}} t \curvearrowleft\curvearrowright s \equiv t \in CC(\mathcal{E})$$

Beweis:

Korrektheit ($s \equiv t \in \mathcal{E}_i \curvearrowright s \equiv_{\mathcal{E}} t$):

I.A.: $s \equiv t \in \mathcal{E}_0 = \mathcal{E} \cup R$
$\curvearrowright s \equiv_{\mathcal{E}} t$ (Reflexivität von $\equiv_{\mathcal{E}}$)
I.S.: $s \equiv t \in \mathcal{E}_{i+1} = \mathcal{E}_i \cup S(\mathcal{E}_i) \cup T(\mathcal{E}_i) \cup C(\mathcal{E}_i)$
$\curvearrowright s \equiv_{\mathcal{E}} t$ ($\equiv_{\mathcal{E}}$ Kongruenzrelation)

Vollständigkeit ($s \equiv_{\mathcal{E}} t \curvearrowright \exists i : s \equiv t \in \mathcal{E}_i$):

Die Vollständigkeit beruht auf dem Satz 3.1 (von Birkhoff). Im Beweis für Satz 3.3 ist dies enthalten.

q.e.d.

Bisher haben wir noch kein Beweisverfahren vorgestellt, dass ein komplettes Entscheidungsverfahren ist. Das Problem in dem zuletzt vorgestellten ist, dass man nicht wissen kann, wann man mit einer Iteration aufhören und sichergehen kann, dass man eine erschöpfenden Suche vollzogen hat. Im Folgenden werden wir aber zeigen, dass es genügt sich auf die (Teil-) Terme zu bechränken, die in $\mathcal{E}$, s oder t auftreten. Dazu definieren wir zunächst den Begriff des Teilterms (*subterm*).

Definition 3.12 (Teiltermmenge) *Zu jedem Term s sei*

$$Subterms(s) := \{s|_\pi \mid \pi \in Occ(s)\}$$

*die Menge seiner **Teilterme**. Zu jedem Gleichungssystem $\mathcal{E}$ sei*

$$Subterms(\mathcal{E}) := \bigcup_{u \equiv v \in \mathcal{E}} Subterms(u) \cup Subterms(v)$$

Beispiel:

$s := f(m)$, $t := g(k)$, $\mathcal{E} = \{m \equiv g(l), \ldots\}$ (wie oben)

- $Subterms(s) = \{m, f(m)\}$
- $Subterms(t) = \{k, g(k)\}$
- $Subterms(\mathcal{E}) = \{m, g(l), l, k, i, j, f(j), f(i), g(k)\}$

Beachte:

Wenn $|\mathcal{E}| = n < \infty$, dann ist auch $|Subterms(\mathcal{E})| = m < \infty$.

Beschränkt man sich nur noch auf die Terme in $S := Subterms(s) \cup Subterms(t) \cup Subterms(\mathcal{E})$ zum Nachweis von $s \equiv_{\mathcal{E}} t$, so erhält man einen eingeschränkten Kongruenzabschluss, der in endlich vielen Schritten berechnet werden kann.

Definition 3.13 (Kongruenzabschluss bezüglich einer Menge von Termen) *Sei $\mathcal{E}$ eine Menge von Grundidentitäten über Σ und $s,t \in \mathcal{T}(\Sigma,\mathcal{V})$. Sei ferner $S := Subterms(s) \cup Subterms(t) \cup Subterms(\mathcal{E})$. Wir definieren:*

$$\mathcal{E}_0^S := (\mathcal{E} \cup R) \cap S \times S$$

$$\mathcal{E}_{i+1}^S := \left(\mathcal{E}_i^S \cup S(\mathcal{E}_i^S) \cup T(\mathcal{E}_i^S) \cup C(\mathcal{E}_i^S)\right) \cap S \times S$$

*Der **Kongruenzabschluss** bezüglich S ist definiert durch:*

$$CC^S(\mathcal{E}) := \bigcup_{i \in \mathbb{N}} \mathcal{E}_i^S$$

Beispiel:

($\mathcal{E}, s, t$ wie oben)

- $\mathcal{E}_0^S = \{m \equiv g(l), \ldots, f(i) \equiv g(k), m \equiv m, \ldots\}$
- $\mathcal{E}_1^S = \mathcal{E}_0^S \cup \underbrace{\{g(l) \equiv m, \ldots, g(k) \equiv f(i)\}}_{\text{Symmetrie}} \cup \underbrace{\{i \equiv f(j)\}}_{\text{Transitivität}} \cup \underbrace{\{g(k) \equiv g(l), f(i) \equiv f(j)\}}_{\text{Kongruenz}}$

$|S| = 10 \curvearrowright |S \times S| = 100 \curvearrowright |\mathcal{E}_i^S| \leq 100$

Lemma 3.5 *Sei $\mathcal{E}$ eine endliche Menge von Grundidentitäten, $s, t \in \mathcal{T}(\Sigma)$, $S := Subterms(s) \cup Subterms(t) \cup Subterms(\mathcal{E})$. Dann existiert ein $i \in \mathbb{N}$ mit $\mathcal{E}_i^S = \mathcal{E}_{i+1}^S$. Daraus folgt $CC^S(\mathcal{E}) = \mathcal{E}_i^S$.*

Beweis:

Aus $\mathcal{E}_i^S \subseteq \mathcal{E}_{i+1}^S$ und der Endlichkeit von $S \times S$ folgt das Lemma sofort.

q.e.d.

Entscheidungsverfahren für das Wortproblem bei Grundidentitäten

(1) Konstruiere $\mathcal{E}_0^S, \mathcal{E}_1^S, \mathcal{E}_2^S, \ldots$ bis ein $\mathcal{E}_i^S$ erreicht ist mit $\mathcal{E}_i^S = \mathcal{E}_{i+1}^S$.

(2) Wenn $s \equiv t \in \mathcal{E}_i^S$, dann gilt $s \equiv_{\mathcal{E}} t$, andernfalls nicht.

Satz 3.3 *Sei $\mathcal{E}$ eine Menge von Grundidentitäten, $s, t \in \mathcal{T}(\Sigma)$, $S := Subterms(s) \cup Subterms(t) \cup Subterms(\mathcal{E})$. Dann gilt:*

$$s \equiv_{\mathcal{E}} t \curvearrowleft\!\!\curvearrowright s \equiv t \in CC^S(\mathcal{E})$$

Beweis:

Korrektheit ($s \equiv t \in CC^S(\mathcal{E}) \curvearrowright s \equiv_{\mathcal{E}} t$):

Folgt aus der Korrektheit von $CC(\mathcal{E})$, denn $CC^S(\mathcal{E}) \subset CC(\mathcal{E})$

Vollständigkeit ($s \equiv_{\mathcal{E}} t \curvearrowright s \equiv t \in CC^S(\mathcal{E})$):

Wegen des Satzes 3.1 (von Birkhoff) reicht es zu zeigen:
$\forall n \in \mathbb{N} : s \leftrightarrow_{\mathcal{E}}^n t \curvearrowright s \equiv t \in CC^S(\mathcal{E})$

I.A.: $s \leftrightarrow_{\mathcal{E}}^0 t \curvearrowright s = t \curvearrowright s \equiv t \in R \curvearrowright s \equiv t \in \mathcal{E}_0^S$,
denn: $Subterms(s) \cup Subterms(t) \subseteq S$.

I.S.:

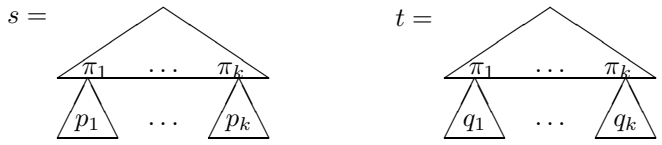
(1) In der Herleitung von $s \leftrightarrow_{\mathcal{E}}^{n} t$ wird mindestens einmal eine Gleichung aus $\mathcal{E}$ an der obersten Stelle ε angewendet.

Daraus folgt: es existiert $u \equiv v \in \mathcal{E}$ oder $v \equiv u \in \mathcal{E}$ und ein $k < n$ mit $s = s_0 \leftrightarrow_{\mathcal{E}} s_1 \leftrightarrow_{\mathcal{E}} \ldots \leftrightarrow_{\mathcal{E}} s_{k-1} \leftrightarrow_{\mathcal{E}} \underbrace{s_k}_{=u} \leftrightarrow_{\mathcal{E}} \underbrace{s_{k+1}}_{=v} \leftrightarrow_{\mathcal{E}} \ldots \leftrightarrow_{\mathcal{E}} s_n = t$.

D. h.: $s \leftrightarrow_{\mathcal{E}}^{k} u$ und $v \leftrightarrow_{\mathcal{E}}^{n-k-1} t$.
Wenn $Subterms(s) \cup Subterms(u) \subseteq S$, dann $s \equiv u \in CC^S(\mathcal{E})$.
Wenn $Subterms(v) \cup Subterms(t) \subseteq S$, dann $v \equiv t \in CC^S(\mathcal{E})$.
$Subterms(u), Subterms(v) \subseteq S$, weil $u \equiv v \in \mathcal{E}$ und $Subterms(\mathcal{E}) \subseteq S$.
Das bedeutet: $s \equiv u$, $v \equiv t$, $u \equiv v \in CC^S(\mathcal{E}) \curvearrowright s \equiv t \in CC^S(\mathcal{E})$ (wegen Transitivität)

Wir haben also gezeigt $(*)$:
$Subterms(s) \subseteq S$ und $Subterms(t) \subseteq S$ sowie $s \leftrightarrow_{\mathcal{E}}^{l} t$, $l \leq n$. In der Herleitung wird eine Gleichung an oberster Stelle benutzt $\curvearrowright s \equiv t \in CC^S(\mathcal{E})$.

(2) In der Herleitung von $s \leftrightarrow_{\mathcal{E}}^{n} t$ wird keine Gleichung aus $\mathcal{E}$ an der obersten Stelle ε angewendet.



Mit $p_1 \leftrightarrow_{\mathcal{E}}^{\leq n} q_1, \ldots, p_k \leftrightarrow_{\mathcal{E}}^{\leq n} q_k$

Seien $\pi_1, \ldots, \pi_k$ mit $\pi_i \perp \pi_j$ für $i \neq j$ die *obersten* Stellen von s, an denen jeweils Gleichungen angewendet werden.
$\curvearrowright s|_{\pi_i} = p_i$, $p_i \leftrightarrow_{\mathcal{E}}^{\leq n} q_i$, $t = s[q_1]_{\pi_1} \cdots [q_k]_{\pi_k}$
Es gilt: $\underbrace{Subterms(p_i)}_{\subseteq Subterms(s)} \subseteq S$ und $\underbrace{Subterms(q_i)}_{\subseteq Subterms(t)} \subseteq S$ sowie
$p_i \leftrightarrow_{\mathcal{E}}^{\leq n} q_i$, wobei in der Herleitung Gleichungen an oberster Stelle benutzt werden.
$\overset{(*)}{\curvearrowright} p_1 \equiv q_1, \ldots, p_k \equiv q_k \in CC^S(\mathcal{E})$
$\curvearrowright s \equiv t \in CC^S(\mathcal{E})$, da $CC^S(\mathcal{E})$ ist unter eingeschränkter Kongruenz abgeschlossen ist.

q.e.d.

Korollar 3.1 (Entscheidbarkeit des Wortproblems für Grundidentitäten) *Sei $\mathcal{E}$ eine Menge von Grundidentitäten. Dann ist das Wortproblem über $\mathcal{E}$ entscheidbar.*

Bei der oben angegebenen Lösung des Wortproblems bei Grundidentitäten kann man schnell die Übersicht verlieren. Daher ist es geschickter, äquivalente Terme (deren Äquivalenz aus einem Gleichungssystem folgt) in einer Äquivalenzklasse zu repräsentieren.

(1) Äquivalenzschritt:
Alle nicht-disjunkten Mengen werden miteinander vereinigt.

(2) Kongruenzschritt:
Für alle äquivalenten Terme $s_1, t_1/s_2, t_2/\ldots/s_n, t_n$ bilde $\{f(s_1,\ldots,s_n), f(t_1,\ldots,t_n)\}$, falls $f(s_1,\ldots,s_n), f(t_1,\ldots,t_n) \in S$. Sobald sich nichts mehr ändert, stoppe den Algorithmus.

Folgende Iteration ist zur *praktischen* Berechnung von $CC^S(\mathcal{E})$ geeigneter:

Sei $\Rightarrow_\mathcal{E}$ definiert als $s \Rightarrow_\mathcal{E} t :\curvearrowright\curvearrowright s \equiv t \in \mathcal{E}$. Wie üblich ist $\Leftrightarrow^*_\mathcal{E}$ die transitiv-reflexiv-symmetrische Hülle von $\Rightarrow_\mathcal{E}$. Sei $Aq(\mathcal{E}) := \{s \equiv t \mid s \Leftrightarrow^*_\mathcal{E} t\}$.
Dann verwenden wir die folgende Iteration:

- $\mathcal{E}_0^{S'} := Aq(\mathcal{E}) \cap S \times S$
- $\mathcal{E}_{i+1}^{S'} := Aq\Big(\mathcal{E}_i^{S'} \cup C\big(\mathcal{E}_i^{S'}\big)\Big) \cap S \times S$

Wir repräsentieren die Äquivalenzrelation als *Quotientenmenge*:

- $S_0 := S/_{\Rightarrow_{\mathcal{E}_0^{S'}}} = \{[s]_{\mathcal{E}_0^{S'}} \mid s \in S\}$
- S_{i+1} analog

Es gilt: $\bigcup_{i\in\mathbb{N}} \mathcal{E}_i^{S'} = \bigcup_{i\in\mathbb{N}} \mathcal{E}_i^S = CC^S(\mathcal{E})$

Algorithmus:
KONGRUENZABSCHLUSS$(\mathcal{E}, s, t)$

Eingabe: endliche Menge $\mathcal{E}$ von Grundidentitäten, zwei Grundterme s, t

Ausgabe: "true", falls $s \equiv_\mathcal{E} t$, sonst "false"

Vorgehen:

(1) Berechne $S := Subterms(\mathcal{E}) \cup Subterms(s) \cup Subterms(t)$

(2) Sei $L := \big\{\{u, v\} \mid u \equiv v \in \mathcal{E}\big\} \cup \big\{\{w\} \mid w \in S\big\}$

(3) Vereinige alle Mengen M_1, M_2 mit $M_1 \cap M_2 \neq \emptyset$

(4) Sei $K := L \cup \big\{\{f(s_1,\ldots,s_n), f(t_1,\ldots,t_n)\} \mid f \in \Sigma,$
$\exists M_i \in L : s_i, t_i \in M_i, f(s_1,\ldots,s_n), f(t_1,\ldots,t_n) \in S\big\}$

(5) Vereinige alle Mengen $M_1, M_2 \in K$ mit $M_1 \cap M_2 \neq \emptyset$

(6) Falls $K \neq L$, dann setze $L := K$ und gehe zu Schritt (4)

(7) Falls es eine Menge $M \in L$ gibt mit $s, t \in M$, dann gib aus "true", ansonsten "false".

Satz 3.4 *Der Algorithmus* KONGRUENZABSCHLUSS *terminiert immer und ist korrekt.*

Wir fassen zusammen:

- Bei Grundidentitäten kann man durch den *Kongruenzabschluss* das *Wortproblem* lösen und muss dabei nur Terme aus S betrachten.
- Beim Lösen des Wortproblems bei Grundidentitäten mit $\leftrightarrow^*_\mathcal{E}$ ist dies nicht immer der Fall, da bei Anwendung einer Gleichung im Inneren eines Terms der Rest des Terms erhalten bleibt.

- Bei Termen mit Variablen ist die Einschränkung auf S erst recht unbrauchbar, d. h. wir benötigen andere Techniken, wenn wir Variablen behandeln wollen.

3.3 Termersetzungssysteme

In diesem Abschnitt ist unser Ziel das Lösen des Wortproblems für $\mathcal{E}, s, t$ mit Variablen. Dabei ist die Intuition zu zeigen, ob $s \equiv_{\mathcal{E}} t$ für alle Variablenbelegungen gilt. Aufgrund des *Indeterminismus* von $\leftrightarrow^*_{\mathcal{E}}$ hatten wir bisher das Problem, dass wir das Wortproblem nur *semi-entscheiden* können und dies meistens mit einem sehr hohem Aufwand. Daher wollen wir im Folgenden den Indeterminismus so weit wie möglich reduzieren. Ein erster Schritt dahin ist, dass wir Gleichungen nur noch in eine Richtung anwenden werden.

Definition 3.14 (Termersetzungssystem) *Für $l, r \in \mathcal{T}(\Sigma, \mathcal{V})$ heißt $l \to r$ **Regel** über Σ und $\mathcal{V}$, falls*

- $\mathcal{V}(r) \subseteq \mathcal{V}(l)$
- $l \notin \mathcal{V}$

*Eine Menge $\mathcal{R}$ von Regeln heißt **Termersetzungssystem (TES)** (term rewriting system). Für jedes TES $\mathcal{R}$ ist die **(Term-)Ersetzungsrelation** $\to_{\mathcal{R}} \subseteq \mathcal{T}(\Sigma, \mathcal{V}) \times \mathcal{T}(\Sigma, \mathcal{V})$ definiert durch*

$$s \to_{\mathcal{R}} t :\curvearrowleft\curvearrowright s|_\pi = l\sigma \wedge t = s[r\sigma]_\pi$$

für $\pi \in Occ(s)$, $l \to r \in \mathcal{R}$ und $\sigma \in SUB(\Sigma, \mathcal{V})$.

*$s \to_{\mathcal{R}} t$ heißt ein **(Term-)Ersetzungsschritt**, bei dem s an der Stelle π **reduziert** wird. Jeder Teilterm $s|_\pi$, auf den die linke Seite einer Regel matcht, heißt **Redex** (reducible expression).*

Wir schreiben statt $s \to_{\mathcal{R}} t$ auch $s \to t$ oder $t \leftarrow_{\mathcal{R}} s$. Wie bei Gleichungssystemen beschränken wir uns auf endliche TESe.

Während wir bei Gleichungssystemen an der $\leftrightarrow^*$-Relation interessiert waren, gilt unser Augenmerk nun der $\to^*$-Relation. Ein Termersetzungssystem kann man auch als funktionales Programm betrachten.

Beispiel:

$$\mathcal{R} := \{ plus(\mathcal{O}, y) \to y,\ plus(succ(x), y) \to succ\big(plus(x, y)\big) \}$$

$$\begin{aligned} plus\Big(succ\big(succ(\mathcal{O})\big), succ(\mathcal{O})\Big) &\to_{\mathcal{R}} succ\Big(plus\big(succ(\mathcal{O}), succ(\mathcal{O})\big)\Big) \\ &\to_{\mathcal{R}} succ\Big(succ\big(plus(\mathcal{O}, succ(\mathcal{O}))\big)\Big) \\ &\to_{\mathcal{R}} succ\Big(succ\big(succ(\mathcal{O})\big)\Big) \end{aligned}$$

Um das Wortproblem für ein Gleichungssystem $\mathcal{E}$ zu lösen, widmen wir uns jetzt der Aufgabe ein zu $\mathcal{E}$ *äquivalentes Termersetzungssystem* $\mathcal{R}$ zu finden.

Definition 3.15 (Äquivalenz von $\mathcal{R}$ zu $\mathcal{E}$) *Sei $\mathcal{E}$ ein Gleichungssystem, $\mathcal{R}$ ein TES. Dann ist $\mathcal{R}$ **äquivalent** zu $\mathcal{E}$:$\curvearrowleft\curvearrowright$ $\leftrightarrow^*_{\mathcal{R}} = \leftrightarrow^*_{\mathcal{E}}$*

Satz 3.5 *$\mathcal{R}$ ist äquivalent zu $\mathcal{E}$:$\curvearrowright$*

- *$l \leftrightarrow^*_{\mathcal{E}} r$ für alle Regeln $l \to r \in \mathcal{R}$ ($\mathcal{R}$ ist **korrekt** für $\mathcal{E}$)*
- *$s \leftrightarrow^*_{\mathcal{R}} t$ für alle Gleichungen $s \equiv t \in \mathcal{E}$ ($\mathcal{R}$ ist **adäquat** für $\mathcal{E}$)*

Beispiel:

- Sei $\mathcal{E}_1 := \{plus(0, y) \equiv y, \ldots\}$
  Dann ist $\mathcal{R}_1 := \{plus(0, y) \to y, \ldots\}$ äquivalent zu $\mathcal{E}_1$
- $\mathcal{E}_2 := \{b \equiv c, b \equiv a, f(a) \equiv f(f(a))\}$
  Dann ist $\mathcal{R}_2 := \{c \to b, a \to b, f(f(b)) \to f(b)\}$
  korrekt für $\mathcal{E}_2$:
  - $c \leftarrow_{\mathcal{E}} b$
  - $a \leftarrow_{\mathcal{E}} b$
  - $f(f(b)) \to_{\mathcal{E}} f(f(a)) \leftarrow_{\mathcal{E}} f(a) \leftarrow_{\mathcal{E}} f(b)$

  adäquat für $\mathcal{E}_2$:
  - $b \leftarrow_{\mathcal{R}} c$
  - $b \leftarrow_{\mathcal{R}} a$
  - $f(a) \leftarrow_{\mathcal{R}} f(b) \leftarrow_{\mathcal{R}} f(f(b)) \leftarrow_{\mathcal{R}} f(f(a))$

Unser Ziel ist es nun, dass zur Überprüfung von $s \leftrightarrow^*_{\mathcal{R}} t$ nicht mehr Regeln beliebig in beide Richtungen angewandt werden sollen. Eine Idee zum Erreichen dieses Ziels ist: Man reduziere s und t solange wie möglich mit $\to_{\mathcal{R}}$:

$$s \to_{\mathcal{R}} s_1 \to_{\mathcal{R}} \ldots \to_{\mathcal{R}} s_n \stackrel{?}{=} t_m \leftarrow_{\mathcal{R}} \ldots \leftarrow_{\mathcal{R}} t_1 \leftarrow_{\mathcal{R}} t$$

Und falls $s_n = t_m$ gilt, so gilt $s \leftrightarrow^*_{\mathcal{R}} t$. Die Reduktionen können dabei beliebig gewählt werden.

Beispiel:

$\mathcal{R}$ sei das "*plus*-TES" und $\mathcal{E}$ das "*plus*-Gleichungssystem"

Frage: Gilt $plus\Big(succ\big(succ(\mathcal{O})\big), x\Big) \equiv_{\mathcal{E}} plus\big(succ(\mathcal{O}), succ(x)\big)$?

Dazu: Gilt $plus\Big(succ\big(succ(\mathcal{O})\big), x\Big) \leftrightarrow^*_{\mathcal{R}} plus\big(succ(\mathcal{O}), succ(x)\big)$?

$$\begin{aligned} plus\Big(succ\big(succ(\mathcal{O})\big), x\Big) &\stackrel{(2)}{\to}_{\mathcal{R}} succ\big(plus(succ(\mathcal{O}), x)\big) \\ &\stackrel{(2)}{\to}_{\mathcal{R}} succ\Big(succ\big(plus(\mathcal{O}, x)\big)\Big) \\ &\stackrel{(1)}{\to}_{\mathcal{R}} succ\big(succ(x)\big) \end{aligned}$$

$$\begin{aligned} plus\big(succ(\mathcal{O}), succ(x)\big) &\stackrel{(2)}{\to}_{\mathcal{R}} succ\Big(plus\big(\mathcal{O}, succ(x)\big)\Big) \\ &\stackrel{(1)}{\to}_{\mathcal{R}} succ\big(succ(x)\big) \end{aligned}$$

$\rightsquigarrow$ "true"

Das Verfahren ist korrekt, wenn "true" ausgegeben wird, denn:

$$s \to^*_{\mathcal{R}} s_n = t_m \leftarrow^*_{\mathcal{R}} t \quad \curvearrowright \quad s \leftrightarrow^*_{\mathcal{R}} t$$

Aber:

(1) Die Auswertung könnte nicht terminieren:
$\mathcal{E} := \{b \equiv c, b \equiv a, f(a) \equiv f(f(a))\}$
$\mathcal{R}_1 := \{b \to c, b \to a, f(a) \to f(f(a))\}$

$$\underbrace{f(a)}_{\text{unendl. Reduktion}} \equiv_{\mathcal{E}} \underbrace{f(c)}_{\text{irreduzibel}} \quad \rightsquigarrow \text{keine Terminierung}$$

Um das zu verhindern verlangen wir, dass $\mathcal{R}$ **terminiert**, d. h. es existieren keine unendlichen $\to_{\mathcal{R}}$-Folgen.

(2) Das Ergebnis der Auswertung könnte nicht eindeutig sein:
$\mathcal{E} := \{b \equiv c, b \equiv a, f(a) \equiv f(f(a))\}$
$\mathcal{R}_2 := \{b \to c, b \to a, f(f(a)) \to f(a)\}$

$$\underbrace{f(a)}_{\text{irreduzibel}} \equiv_{\mathcal{E}} \underbrace{f(c)}_{\text{irreduzibel}} \quad \rightsquigarrow \text{“false”}$$

Aber: $f(a) \leftarrow_{\mathcal{R}_2} f(b) \to_{\mathcal{R}_2} f(c)$

Also verlangen wir, dass aus $s \leftrightarrow^*_{\mathcal{R}} t$ immer folgt, dass es ein q gibt mit $s \to^*_{\mathcal{R}} q \leftarrow^*_{\mathcal{R}} t$. Man sagt dann: “$\mathcal{R}$ hat die Church-Rosser**-Eigenschaft**”

Da $\mathcal{R}$ terminieren und die Church-Rosser-Eigenschaft haben soll, kann man oft nicht einfach $\equiv$ durch $\to$ oder $\leftarrow$ ersetzen. Wir werden daher in den nächsten Kapiteln kennenlernen, wie man aus einem Gleichungssystem ein Termersetzungssystem mit den geforderten Eigenschaft generiert.

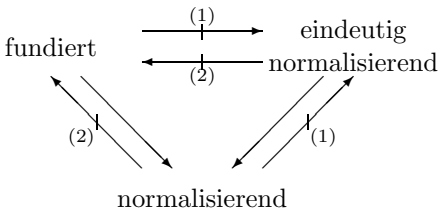
Definition 3.16 (fundierte Relation) *Sei $\to \subseteq M \times M$. Dann heißt $\to$ **fundiert** $:\Leftrightarrow$ es existiert keine undendliche Folge von Elementen $t_0, t_1, \ldots \in M$ mit $t_0 \to t_1 \to \ldots$.*

Beispiel:

- $>_{\mathbb{N}}$ ist fundiert
- $\rhd$ ist fundiert
- $<_{\mathbb{N}}$ ist nicht fundiert, denn:
  $0 < 1 < 2 < \ldots$
- $>_{\mathbb{Z}}$ ist nicht fundiert, denn:
  $1 > -1 > -2 > \ldots$
- $>_{\mathbb{Q}^+}$ ist nicht fundiert, denn:
  $\frac{1}{2} > \frac{1}{4} > \frac{1}{8} > \ldots$
- $>_{\mathbb{N}^*}$ ist fundiert, denn:
  $\pi_1 >_{\mathbb{N}^*} \pi_2 :\Leftrightarrow |\pi_1| >_{\mathbb{N}} |\pi_2|$

Definition 3.17 (Normalform) *Sei $\to \subseteq M \times M$. Ein Objekt $q \in M$ heißt **Normalform** bezüglich $\to :\Leftrightarrow \nexists q' \in M : q \to q'$. q heißt **Normalform eines Objekts** $t \in M :\Leftrightarrow t \to^* q$ und q ist Normalform bezüglich $\to$. Falls die Normalform von t eindeutig ist, so bezeichnet man sie mit $t\!\downarrow$.*
*Die Relation $\to$ heißt **normalsierend**, falls es zu jedem Objekt (mindestens) eine Normalform gibt. Die Relation $\to$ heißt **eindeutig normalsierend**, falls es zu jedem Objekt genau eine Normalform gibt.*

Folgerung (siehe Beispiel unten):



Lemma 3.6 *Jede fundierte Relation ist normalisierend.*

Beweis:

Sei $\rightarrow$ fundiert.
Annahme: $\exists t \in M$ ohne Normalform.
$\curvearrowright t \rightarrow t_1 \rightarrow t_2 \rightarrow \ldots \rightsquigarrow$ Widerspruch zur Annahme

q.e.d.

Definition 3.18 (Terminierung) *Sei $\mathcal{R}$ ein Termersetzungssystem*

$$\mathcal{R} \left\{ \begin{array}{c} \textit{terminiert} \\ \textit{ist normalsierend} \\ \textit{ist eindeutig normalisierend} \end{array} \right\} :\curvearrowright\rightsquigarrow \rightarrow_{\mathcal{R}} \textit{ ist } \left\{ \begin{array}{c} \textit{fundiert} \\ \textit{normalsierend} \\ \textit{eindeutig normalisierend} \end{array} \right\}$$

Beispiel:

(1) $\mathcal{R}_1 := \{a \rightarrow b, a \rightarrow c\}$
$\rightarrow_{\mathcal{R}_1}$ ist fundiert, aber nicht eindeutig normalisierend.

(2) $\mathcal{R}_2 := \{a \rightarrow b, a \rightarrow c, c \rightarrow a\}$
$\rightarrow_{\mathcal{R}_2}$ ist eindeutig normalisierend, aber nicht fundiert.

Die Variablenbedingungen

(1) $\mathcal{V}(r) \subseteq \mathcal{V}(l)$ und

(2) $l \notin \mathcal{V}$

für alle Regeln $l \rightarrow r$ sind notwendig zur Terminierung, denn wenn

(1) $\mathcal{V}(r) \not\subseteq \mathcal{V}(l)$,
also z. B. $f(x) \rightarrow f(y)$ eine Regel ist, dann ist y mit der linken Seite instanzierbar: $f(x) \rightarrow f(f(x)) \rightarrow f(f(f(x))) \rightarrow \ldots$

allgemein: Wenn $\sigma = \{y/l\}$ und $y \in \mathcal{V}(r) \setminus \mathcal{V}(l)$,
dann $l \rightarrow r\sigma[l]_\pi = r\sigma[r\sigma[l]_\pi]_\pi \rightarrow \ldots$

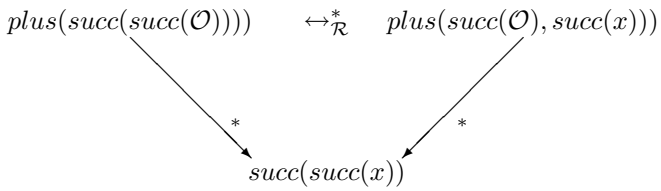
(2) $l \in \mathcal{V}$, also z. B. $x \rightarrow f(x)$ eine Regel ist, dann: $x \rightarrow f(x) \rightarrow f(f(x)) \rightarrow \ldots$

allgemein: Wenn $x \rightarrow r$ und $\sigma_1 = \{x/r\}$, $\sigma_2 = \{x/r\sigma_1\}$, $\ldots$,
dann $x \rightarrow r = x\sigma_1 \rightarrow r\sigma_1 = x\sigma_2 \rightarrow r\sigma_2 = x\sigma_3 \rightarrow \ldots$

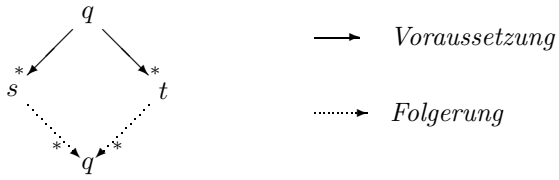
Definition 3.19 (Zusammenführbarkeit, CHURCH-ROSSER-Eigenschaft) *Sei $\rightarrow \subseteq M \times M$. Zwei Elemente $s, t \in M$ sind **zusammenführbar** (joinable) ($s \downarrow t$) $:\Leftrightarrow \exists q \in M : s \rightarrow^* q \leftarrow^* t$. Die Relation $\rightarrow$ hat die CHURCH-ROSSER-**Eigenschaft** $:\Leftrightarrow \forall s, t \in M : s \leftrightarrow^* t \curvearrowright s \downarrow t$. Ein TES $\mathcal{R}$ hat die CHURCH-ROSSER-**Eigenschaft** $:\Leftrightarrow \rightarrow_{\mathcal{R}}$ hat die CHURCH-ROSSER-Eigenschaft.*

Beispiel:

- $\mathcal{R}_1 := \{b \rightarrow c, b \rightarrow a\}$ hat die CHURCH-ROSSER-Eigenschaft nicht.
- Das *plus*-TES hat die CHURCH-ROSSER-Eigenschaft,
  z. B. gilt:



Definition 3.20 (Konfluenz) *Eine Relation $\rightarrow$ über M heißt **konfluent** $:\curvearrowright$ Wenn für alle $s, t, p \in M$ gilt: $p \rightarrow^* s$ und $p \rightarrow^* t$, dann existiert ein $q \in M$ mit $s \rightarrow^* q$ und $t \rightarrow^* q$.*



Satz 3.6 *Für alle Relationen $\rightarrow$ gilt: $\rightarrow$ hat die CHURCH-ROSSER-Eigenschaft $\curvearrowright$ $\rightarrow$ ist konfluent.*

Beweis:

"$\curvearrowright$": $\rightarrow$ habe die CHURCH-ROSSER-Eigenschaft.
Sei $s \leftarrow^* p \rightarrow^* t \curvearrowright s \leftrightarrow^* t \overset{CR}{\curvearrowright} s \downarrow t$
d. h. $s \rightarrow^* q \leftarrow^* t$ für ein $q \in M$

"$\curvearrowleft$": $\rightarrow$ sei konfluent und $s \leftrightarrow^n t$ mit $n \in \mathbb{N}$
I.A.: $s \leftrightarrow^0 t \curvearrowright s = t$ ✓
I.S.: $s \leftrightarrow^{n-1} s' \leftrightarrow t$
$\searrow^* \swarrow^*$ $\quad \exists q : s \rightarrow^* q \leftarrow^* s'$
q

(1) $t \rightarrow s'$:
$s \leftrightarrow^{n-1} s' \leftrightarrow t \quad \curvearrowright s \rightarrow^* q \leftarrow^* s' \leftarrow t \curvearrowright s \downarrow t$
$\searrow^* \swarrow^*$
q

(2) $\underline{s' \to t}$:
$s \leftrightarrow^{n-1} s' \leftrightarrow t \quad$ und $\quad t \to^* p$
$\searrow^* \swarrow^*$
q
$s \to^* q \to^* p \leftarrow^* t \curvearrowright s \downarrow t$

q.e.d.

Lemma 3.7 *Es gilt:*

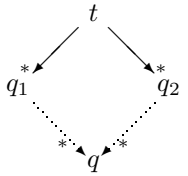
(1) Wenn die Relation $\to$ konfluent ist, dann hat jedes Objekt maximal eine Normalform.

(2) Wenn die Relation $\to$ konfluent und normalisierend ist, dann hat jedes Objekt genau eine Normalform.

(3) Jede eindeutig normalisierende Relation ist konfluent.

Beweis:

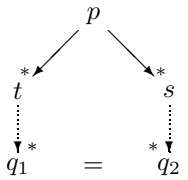
(1) Sei t ein Objekt mit Normalformen q_1 und q_2.



Da q_1 und q_2 Normalformen sind gilt:
$q_1 = q = q_2$

(2) Wegen (1) hat jedes Objekt maximal eine Normalform. Da $\to$ nach Voraussetzung normalsierend ist, hat jedes Objekt mindestens eine Normalform.
$\curvearrowright$ jedes Objekt hat genau eine Normalform.

(3) Sei



Normalformen von s und t: $s\downarrow = q_1$, $t\downarrow = q_2$
$\curvearrowright$ q_1, q_2 sind Normalformen von p
Wegen der eindeutigen Normalisierung gilt:
$q_1 = q_2 = p\downarrow$
$\curvearrowright$ $s \downarrow t$

q.e.d.

Satz 3.7 *Sei $\to \subseteq M \times M$ normalsierend und konfluent, $s, t \in M$. Dann gilt:*

$$s \leftrightarrow^* t \curvearrowleft\curvearrowright s\downarrow = t\downarrow$$

Beweis:

"$\curvearrowright$": $s\downarrow = t\downarrow \curvearrowright s \downarrow t \curvearrowright s \leftrightarrow^* t$

"$\curvearrowleft$": $s \leftrightarrow^* t \overset{\text{Konfl.}}{\curvearrowright} s \downarrow t$:
$s \to^* q \leftarrow^* t \curvearrowright q \to^* q'$ für ein $q \in M$
$\overset{\text{Norm.}}{\curvearrowright}$ $\exists q'$: q' ist Normalform von q
$\curvearrowright$ s und t haben Normalform q'
$\overset{\text{L 3.7}}{\curvearrowright}$ $s\downarrow = q' = t\downarrow$

q.e.d.

Wir fordern nun für Termersetzungssysteme, die das Wortproblem von Gleichungssystemen lösen sollen, *Fundiertheit und Konfluenz*. Termersetzungssysteme, die beide Eigenschaften erfüllen heißen dann *konvergent*. Unser weiteres Ziel wird es sein, solche Termersetzungssysteme automatisch zu konstruieren.

Definition 3.21 (Konvergenz) *Ein TES $\mathcal{R}$ heißt* ***konvergent*** *(oder kanonisch)* $:\curvearrowright\curvearrowright$ *$\mathcal{R}$ terminiert und ist konfluent.*

Beispiel:

Beim *plus*-TES ist "$\downarrow_{\mathcal{R}}$" ein *Interpreter* für funktionale Sprachen.
Sei $t = plus(succ(succ(\mathcal{O})), succ(\mathcal{O}))$, dann berechnet der Interpreter:
$t\downarrow_{\mathcal{R}} = succ(succ(succ(\mathcal{O})))\widehat{=}\ 3$.

Konvergente TESe können nicht nur als Interpreter, sondern auch als *automatische Beweiser* eingesetzt werden.

Algorithmus:
WORTPROBLEM$(\mathcal{R}, s, t)$

Eingabe: konvergentes TES $\mathcal{R}$ äquivalent zu $\mathcal{E}$, zwei Terme s, t

Ausgabe: "true", falls $s \equiv_{\mathcal{E}} t$, sonst "false"

Vorgehen:

(1) Reduziere s und t auf beliebige Weise solange wie möglich. So entstehen die Normalformen $s\downarrow$ und $t\downarrow$.

(2) Falls $s\downarrow = t\downarrow$ gib "true" aus, ansonsten "false".

Satz 3.8 *Es gilt:*

(1) Der Algorithmus WORTPROBLEM *terminiert.*

(2) Der Algorithmus WORTPROBLEM *ist korrekt.*

(3) Wenn zu einem Gleichungssystem $\mathcal{E}$ ein äquivalentes und konvergentes Termersetzungssystem $\mathcal{R}$ existiert, so ist das Wortproblem über $\mathcal{E}$ entscheidbar.

Beweis:

(1) Die Terminierung folgt aus der Terminierung von $\mathcal{R}$ (jede Reduktionsfolge ist endlich)

(2) Aus der Fundiertheit von $\rightarrow_{\mathcal{R}}$ folgt die Normalisierung von $\rightarrow_{\mathcal{R}}$. Daher ist Satz 3.7 anwendbar:
$s\downarrow_{\mathcal{R}} = t\downarrow_{\mathcal{R}} \overset{S3.7}{\curvearrowright\curvearrowright} s \leftrightarrow^*_{\mathcal{R}} t \overset{\mathcal{R}\sim\mathcal{E}}{\curvearrowright\curvearrowright} s \leftrightarrow^*_{\mathcal{E}} t \overset{S3.1}{\curvearrowright\curvearrowright} s \equiv_{\mathcal{E}} t$

(3) Die Entscheidbarkeit folgt direkt aus (2).

q.e.d.

Beispiel:

$\mathcal{E} := \{f\big(x, f(y,z)\big) \equiv f\big(f(x,y), z\big), f(x,e) \equiv x, f\big(x, i(x)\big) \equiv e\}$

Äquivalentes TES $\mathcal{R} :=$

$$\begin{array}{rcl} \{f\big(x, f(y,z)\big) & \to & f\big(f(x,y), z\big) \\ f(x,e) & \to & x \\ f\big(x, i(x)\big) & \to & e\} \end{array}$$

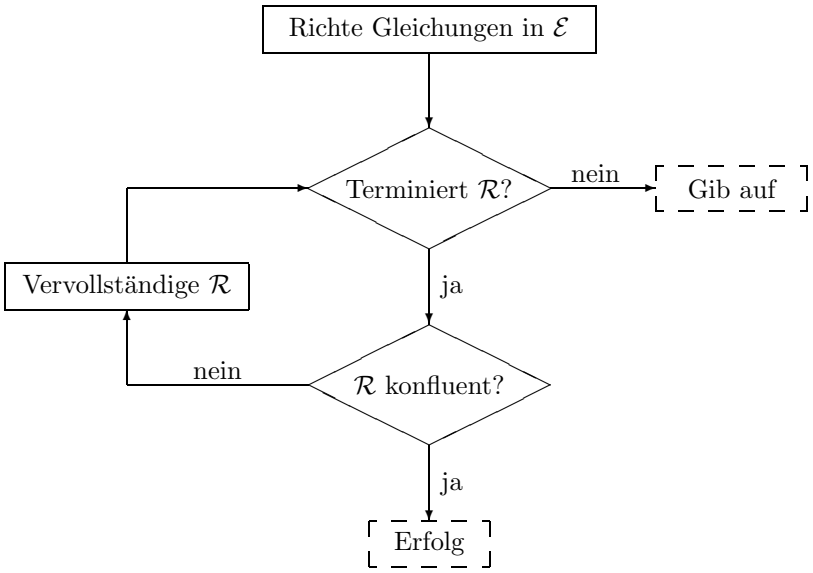
$$\left.\begin{array}{rcl} i(i(x))\downarrow & = & i(i(x)) \\ x\downarrow & = & x \end{array}\right\} \neq \qquad \rightsquigarrow \text{“false”}$$

Die Antwort “false” ist nicht korrekt, weil $\mathcal{R}$ zwar äquivalent zu $\mathcal{E}$ ist und terminiert, aber nicht konfluent ist:

$$f\Big(f\big(x, i(x)\big), i^2(x)\Big) \to f\Big(x, f\big(i(x), i^2(x)\big)\Big) \to f(x,e) \to \overbrace{x}^{NF}$$

$$\underbrace{f\big(e, i^2(x)\big)}_{NF}$$

Wir definieren nun ein Vorgehen, um das Wortproblem zu lösen bzw. automatisch eine **Entscheidungsprozedur für das Wortproblem** zu generieren.



Wenn man bei dieser Prozedur mit $\mathcal{E} = \mathcal{R}_0$ beginnt, und die Vervollständigungen $\mathcal{R}_1, \mathcal{R}_2, \ldots$ liefern, so ergeben sich 3 Möglichkeiten:

(1) Nach n Iterationen erhält man das TES $\mathcal{R}_n$, das konvergent ist – somit endet man mit einem Erfolg.

(2) Man kann die Terminierung von $\mathcal{R}_i$ nicht nachweisen – man endet mit einem Misserfolg. Gegebenenfalls kann man mit einem neuen Versuch (anderes Richten der Gleichungen) starten.

(3) Alle $\mathcal{R}_i$ terminieren, keines ist konfluent: das Verfahren terminiert nicht – der Misserfolg wird nicht bemerkt.

Nach den gezeigten Überlegungen ergibt sich, dass das Wortproblem für ein beliebiges Gleichungssystem *nicht entscheidbar* ist. Jedoch existiert ein Verfahren, das Wortproblem für *konvergente Termersetzungssysteme* zu lösen. Daher wird es die Aufgabe des nächsten Kapitels sein, herauszufinden, ob ein Termersetzungssystem terminiert. Das Kapitel 5 beschäftigt sich dann mit dem Überprüfen der Konfluenz eines Termersetzungssystems. Im 6. Kapitel geht es dann schließlich um die Frage, wie man ein Termersetzungssystem automatisch vervollständigen kann.

Kapitel 4

Terminierung von Termersetzungssystemen

Dieses Kapitel beschäftigt sich mit der Frage, wie man feststellen kann, ob ein Termersetzungssystem terminiert. Die Terminierungsanalyse ist nötig für

- die Überprüfung des Wortproblems
- die Überprüfung der Konfluenz
- genrell wichtige Fragen der Programmanalyse
- Induktionen

Der Terminierungsnachweis entspricht nach Definition 3.18 dem Nachweis, dass bestimmte Relationen fundiert sind.

4.1 NOETHERsche Induktion

Die Induktionen, die wir bisher kennengelernt haben, sind Spezialfälle eines allgemeineren Beweisprinzips, nämlich der NOETHERSCHEN Induktion.

Definition 4.1 (NOETHERsches Induktionsprinzip, EMMY NOETHER) *Sei $\succ$ eine fundierte Relation über einer Menge M, $\varphi(m)$ eine Aussage über Objekte aus M. Für alle $m \in M$ gelte: wenn $\varphi(k)$ für alle $k \in M$ mit $m \succ k$ gilt, dann gilt auch $\varphi(m)$. Dann gilt die Aussage $\varphi(n)$ für alle $n \in M$.*

$$\Big(\forall m \in M : \big(\forall k \in M : m \succ k \curvearrowright \varphi(k)\big) \curvearrowright \varphi(m)\Big) \curvearrowright \forall n \in M : \varphi(n)$$

Satz 4.1 *Das NOETHERsche Induktionsprinzip ist korrekt.*

Beweis:

Annahme: $\forall m \in M : \big(\forall k \in M : m \succ k \curvearrowright \varphi(k)\big) \curvearrowright \varphi(m)$
aber $\exists n_0 \in M : \neg\varphi(n_0)$

Wegen der Annahme gilt: $\underbrace{\big(\forall k \in M : k \succ n_0 \curvearrowright \varphi(k)\big) \curvearrowright \varphi(n_0)}_{(*)}$

Da $\varphi(n_0)$ nicht gilt, folgt $(*)$ kann auch nicht gelten.
$\curvearrowright \exists n_1 \in M : n_0 \succ n_1$ und $\neg\varphi(n_1)$
$\ldots \curvearrowright \exists n_2 \in M : n_1 \succ n_2$ und $\neg\varphi(n_2)$
$\vdots$
$n_0 \succ n_1 \succ n_2 \succ \ldots \quad \rightsquigarrow$ Widerspruch zuer Fundiertheit von $\succ$.

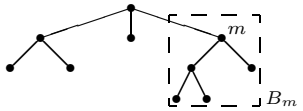
q.e.d.

Satz 4.2 (Lemma von KÖNIG**)** *Ein Baum mit endlichem Verzweigungsgrad, in dem jeder Pfad endlich ist, besitzt nur endlich viele Knoten.*

Beweis:

Sei B ein Baum mit den Voraussetzungen des Satzes 4.2. Sei M die Menge der Knoten von B.

Für alle $m \in M$ sein B_m der Baum, der von m aufgespannt wird (m als Wurzel hat).



Sei $\succ \subseteq M \times M$ mit $m \succ k :\curvearrowright\curvearrowright k$ ist direktes Kind von m. Da es keine unendlichen Pfade gibt, ist $\succ$ fundiert.

Wir werden zeigen: $\forall n \in M : \varphi(n)$, wobei $\varphi(n)$: B_n hat nur endlich viele Knoten.

$$|B_m| = 1 + \underbrace{\sum_{k \prec m} \underbrace{|B_k|}_{<\infty \ (I.V.)}}_{<\infty \ (Vor.)} < \infty$$

$\overset{N.I.}{\curvearrowright}\ \forall n \in M : \varphi(n)$

q.e.d.

4.2 Entscheidbarkeitsresultate zur Terminierung

Termersetzungssysteme sind TURING-vollständige Programmiersprachen, d. h. man kann jede berechenbare Funktion damit programmieren. Mit der Berechenbarkeitstheorie folgt daraus, dass das Halteproblem, und damit das *Wortproblem für Termersetzungssysteme unentscheidbar* ist.

Satz 4.3 *Sei $\mathcal{R}$ ein Termersetzungssystem über Σ und $\mathcal{V}$, sowie $t \in \mathcal{T}(\Sigma, \mathcal{V})$. Das Problem, ob t keine unendliche Reduktion (Auswertung) mit $\rightarrow_{\mathcal{R}}$ hat (**Halteproblem**), ist semientscheidbar. Das Problem, ob $\mathcal{R}$ terminiert (**universelles Halteproblem**), ist unentscheidbar.*

Beweis:

Für jede Turingmaschine TM kann man ein Termersetzungssystem $\mathcal{R}_{TM}$ angeben, das TM simuliert. Die *Semi-Entscheidbarkeit* des Halteproblems kan man folgendermaßen zeigen:

Man erzeuge einen Suchbaum mit Wurzel t und jeder Knoten mit Markierung u hat als Kinder die Knoten $u_1, \ldots, u_n$ für alle $u \rightarrow_{\mathcal{R}} u_i$. Der Suchbaum hat einen endlichen Verzweigungsgrad wegen der Variablenbedingung $\mathcal{V}(r) \subseteq \mathcal{V}(l) \forall\ l \rightarrow r \in \mathcal{R}$.

$\overset{S4.2}{\curvearrowright}$ Der Baum ist endlich $\curvearrowright\curvearrowright$ Der Baum hat keine unendlichen Pfade

$\curvearrowright\curvearrowright$ t terminiert

Die Konstruktion des Suchbaums mit der Terminationsüberprüfung ist *semientscheidbar.*

q.e.d.

Beispiel:

Entscheidbarkeit für TES ohne Variablen auf rechten Regelseiten:

$$\begin{aligned} and(T,T) &\to T \\ and(x,F) &\to F \\ and(F,x) &\to and(T,not(T)) \\ not(F) &\to T \\ not(T) &\to and(F,F) \end{aligned}$$

$$\begin{aligned} & \underline{and(F,F)} \\ \to_{\mathcal{R}} & \; and(T,not(T)) \\ \to_{\mathcal{R}} & \; and(T,\underline{and(F,F)}) \\ \to_{\mathcal{R}} & \; and(T,and(T,not(T))) \\ \to_{\mathcal{R}} & \; \ldots \quad \rightsquigarrow \text{keine Terminierung} \end{aligned}$$

Lemma 4.1 *Sei $\mathcal{R}$ ein Termersetzungssystem mit $\mathcal{V}(r) = \emptyset \; \forall \; l \to r \in \mathcal{R}$. Dann gilt:*

$$\mathcal{R} \text{ terminiert } \curvearrowright\!\!\!\!\!\!\curvearrowleft \nexists \; l \to r \in \mathcal{R} : r \to_{\mathcal{R}}^{+} t \text{ mit } t \trianglerighteq r$$

Beweis:

"$\curvearrowright$": Wenn $r \to_{\mathcal{R}}^{+} t$ und $t|_\pi = r$, dann existiert eine unendliche Reduktion:
$r \to_{\mathcal{R}}^{+} t = t[r]_\pi \to_{\mathcal{R}}^{+} t[t]_\pi = t[t[r]_\pi]_\pi \to_{\mathcal{R}}^{+} \ldots$

"$\curvearrowleft$": Induktion über die Anzahl der Regeln in $\mathcal{R}$:
I.A.: $\mathcal{R} = \emptyset$ ✓
I.V.: $\mathcal{R} \neq \emptyset$
Annahme: $\mathcal{R}$ terminiert nicht
$\curvearrowright$ es existiert ein Term t mit unendl. Reduktion:
$t = t_0 \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} \ldots$
Wegen der Minimalität von t muss irgendwann in der Auswertung an der obersten Position reduziert werden.
$\exists i \in \mathbb{N} : t_i = l\sigma$ und $t_{i+1} = r\sigma$
für $l \to r \in \mathcal{R}$ und $\sigma \in SUB(\Sigma, \mathcal{V})$.
Da $\mathcal{V}(r) = \emptyset \curvearrowright t_{i+1} = r$
Also $t_0 \to_{\mathcal{R}} \ldots \to_{\mathcal{R}} t_{i+1} = r \to_{\mathcal{R}} \ldots$
$\exists \; l \to r \in \mathcal{R}$ mit r hat unendliche Reduktion.
(1) $l \to r$ wird nicht mehr verwendet
$\curvearrowright \mathcal{R} \setminus \{l \to r\}$ terminiert nicht
(2) $l \to r$ wird noch verwendet
$\curvearrowright r \to_{\mathcal{R}}^{*} t_j = t_j[l\sigma']_{\pi'} \to_{\mathcal{R}} t_{j+1} = t_j[r]_\pi$
$\rightsquigarrow$ Widerspruch

q.e.d.

Algorithmus:
RIGHT-GROUND-TERMINIERUNG($\mathcal{R}$)

Eingabe: TES $\mathcal{R}$ mit $\mathcal{V}(r) = \emptyset$ für alle $l \to r \in \mathcal{R}$

Ausgabe: "true", falls $\mathcal{R}$ terminiert, sonst "false"

Vorgehen:

(1) Für $\mathcal{R} = \{l_1 \to r_1, \ldots, l_n \to r_n\}$ sei $T_i := \{r_i\}$ $(1 \leq i \leq n)$

(2) Für alle $1 \leq i \leq n$ setze $T_i := \{t \mid s \in T_i, s \to_{\mathcal{R}} t\}$

(3) Falls alle $T_i = \emptyset$, dann gib "true" aus und brich ab.

(4) Falls es ein i und ein $t \in T_i$ gibt mit $t \trianglerighteq r_i$, dann gib "false" aus und brich ab.

(5) Gehe zu Schritt (2)

Beispiel:

Sei $\mathcal{R}$ das *and*-TES aus dem obigen Beispiel

Schritt	T_1	T_2	T_3	T_4	T_5
(1)	T	F	$and(T, not(T))$	T	$\underline{and(F, F)}$
(2)	$\emptyset$	$\emptyset$	$and(T, and(F, F))$	$\emptyset$	F $and(T, not(T))$
(2)	$\emptyset$	$\emptyset$	$and(T, F)$, $and(T, and(T, not(t)))$	$\emptyset$	$and(T, \underline{and(F, F)})$

$\rightsquigarrow$ "false"

Satz 4.4 *Sei $\mathcal{R}$ ein Termersetzungssystem mit $\mathcal{V}(r) = \emptyset$ für alle $l \to r \in \mathcal{R}$. Dann ist entscheidbar, ob $\mathcal{R}$ terminiert und der Algorithmus* RIGHT-GROUND-TERMINIERUNG *ist ein Entscheidungsverfahren.*

Beweis:

(1) $\mathcal{R}$ terminert.
$\curvearrowright$ Es existiert kein t und keine Regel $l \to r$ mit $r \to^+ t$ und $t \trianglerighteq r$ (L 4.1)
$\curvearrowright$ Der Algorithmus gibt nie "false" aus.
Wegen der Terminierung hat jedes r nur endlich viele Reduktionen. Wegen Satz 4.2 hat jeder Suchbaum für alle r_i nur endlich viele Knoten.
$\curvearrowright$ nach endlich vielen Schritten sind alle $T_i = \emptyset$
$\curvearrowright$ Der Algorithmus terminiert mit "true"

(2) $\mathcal{R}$ terminert nicht.
$\curvearrowright$ Es existiert ein i mit $r_i \to^+ t$ und $t \trianglerighteq r_i$ (L 4.1)
$\curvearrowright$ nach endlich vielen Schritten gilt $t \in T_i$ und der Algorithmus terminiert mit "false"

q.e.d.

4.3 Terminierung mit Reduktionsrelation

Unser vorrangiges Ziel in diesem Kapitel ist es, einen *Terminierungsnachweis für beliebige Termersetzungssysteme* führen zu können. Wegen der Unentscheidbarkeit kann aber jedes automatische Verfahren nicht vollständig sein. Da ein Termersetzungssystem $\mathcal{R}$ genau dann terminiert, wenn seine Ersetzungsrelation $\rightarrow_{\mathcal{R}}$ fundiert ist, ist nun die Idee eine fundierte Relation $\succ$ zu finden, und zu zeigen, dass $\rightarrow_{\mathcal{R}} \subseteq \succ$ gilt; denn damit wäre gezeigt, dass auch $\rightarrow_{\mathcal{R}}$ fundiert ist. Leider gibt es unendlich viele Terme, weshalb es nötig ist ein Verfahren zu entwickeln, welches $\rightarrow_{\mathcal{R}} \subseteq \succ$ überprüft, ohne jeden Term zu überprüfen. Ein Ansatz ist es $l \succ r$ für alle Regeln $l \rightarrow r \in \mathcal{R}$ zu testen. Dies alleine stellt aber nicht sicher, dass $s \rightarrow_{\mathcal{R}} t \curvearrowright s \succ r$ gilt.

Beispiel:

$endless(x) \rightarrow endless(succ(x))$ terminiert nicht.
Aber es existiert eine fundierte Relation $\succ$ mit $endless(x) \succ endless(succ(x))$, und zwar:
$p \succ q :\curvearrowright\curvearrowright p = endless(x)$ und $q = endless(succ(x))$.

Daher fordern wir für die fundierte Relation $\succ$ ferner, dass sie stabil – also unter Substitution abgeschlossen – ist. Ein Beispiel für eine solche Realtion wäre die echte Teilterm-Relation $\rhd$.

Beispiel:

$infty(x) \rightarrow succ(infty(x))$ terminiert nicht. Aber es existiert eine fundierte und stabile Relation $\succ$ mit $infty(x) \succ succ(infty(x))$, und zwar:
$p \succ q :\curvearrowright\curvearrowright p = infty(\ldots)$ und $q = succ(\ldots)$

Dieses zweite Beispiel zeigt uns, dass die *Fundiertheit* und die *Stabilität* immer noch nicht für unsere Zwecke ausreichen. Unsere dritte Forderung ist daher die *Monotonie*. Da wir bisher eine solche Relation noch nicht kennengelernt haben, werden wir nun die *Einbettungsordnung* definieren, die die drei geforderten Eigenschaften besitzt.

Definition 4.2 (Einbettungsordnung) *Für eine Signatur Σ ist die* ***Einbettungsordung*** $\succ_{emb}$ *(embedding order) über $\mathcal{T}(\Sigma, \mathcal{V})$ definiert als $s \succ_{emb} t :\curvearrowright\curvearrowright$*

- $s = f(s_1, \ldots, s_n)$ *und* $s_i \succeq_{emb} t$ *für ein* $i \in \{1, \ldots, n\}$ *oder*
- $s = f(s_1, \ldots, s_n)$, $t = f(t_1, \ldots, t_n)$ *und* $s_i \succ_{emb} t_i$ *für ein* $i \in \{1, \ldots, n\}$ *sowie* $s_j \succeq_{emb} t_j$ *für alle* $j \neq i$

Hierbei ist $\succeq_{emb}$ die reflexive Hülle von $\succ_{emb}$.

Beispiel:

- $infty(succ(x)) \succ_{emb} infty(x)$, denn $succ(x) \succ_{emb} x$, denn $x \succeq_{emb} x$
- $succ(plus(infty(succ(x)), y)) \succ_{emb} plus(infty(x), y)$

Lemma 4.2 *$\succ_{emb}$ ist fundiert, stabil, monoton und transitiv.*

Beweis:

Fundiertheit folgt aus $\succ_{emb} \subseteq \rightarrow_{||}$, denn $\rightarrow_{||}$ ist fundiert.
Beweis davon durch strukturelle Induktion über s:

(1) $s = f(s_1, \ldots, s_n)$, $s_i \succeq_{emb} t$
Wegen I.V. gilt: $|s_i| \geq |t| \curvearrowright |s| > |s_i| \geq |t|$ ✓

(2) $s = f(s_1, \ldots, s_n)$, $t = f(t_1, \ldots, t_n)$ und $s_i \succ_{emb} t_i$, $s_j \succeq_{emb} t_j$ für alle $j \neq i$
$|s| = 1 + \underbrace{|s_1|}_{\geq |t_1|} + \ldots + \underbrace{|s_i|}_{\geq |t_i|} + \ldots + \underbrace{|s_n|}_{\geq |t_n|} > 1 + |t_1| + \ldots + |t_i| + \ldots + |t_n|$

Rest des Beweises analog mit struktureller Induktion

q.e.d.

Definition 4.3 (Reduktionsrelation, -ordung) *Eine Relation $\succ$ über Termen, die fundiert, stabil und monoton ist heißt* ***Reduktionsrelation****. Eine transitive Reduktionsrelation heißt* ***Reduktionsordnung***

Beachte:

Eine Ordnung ist transitiv und antisymmetrisch. Fundierte Relationen sind asymmetrisch und damit antisymmetrisch.

Satz 4.5 (Manna und Ness, 1970) *Ein Termersetzungssystem $\mathcal{R}$ terminiert $\curvearrowright\!\!\!\!\!\curvearrowright$ es gibt eine Reduktionsrelation $\succ$ mit $l \succ r$ für alle $l \rightarrow r \in \mathcal{R}$.*

Beweis:

"$\curvearrowright$" Für alle Terme s, t gilt: $s \rightarrow_{\mathcal{R}} t \curvearrowright s \succ t$,
denn $s|_\pi = l\sigma$, $t = s[r\sigma]_\pi$
$\curvearrowright l\sigma \succ r\sigma$ (Stabilität) und $\underbrace{s[l\sigma]_\pi}_{s} \succ \underbrace{s[r\sigma]_\pi}_{t}$ (Monotonie)
Die Terminierung folgt aus der Fundiertheit von $\succ$

"$\curvearrowleft$" Sei $\succ = \rightarrow_{\mathcal{R}}$
$\succ$ ist fundiert, weil $\mathcal{R}$ terminiert und stabil und monoton wegen Lemma 3.4
Außerdem gilt $l \rightarrow_{\mathcal{R}} r$ für alle Regeln.

q.e.d.

Der Satz 4.5 ist sowohl hinreichendes als auch notwendiges Kriterium für die Terminierung eines Termersetzungssystems. Trotzdem bleibt die Terminierung von Termersetzungssystemen *unentscheidbar*, weil das Finden einer Reduktionsrelation unentscheidbar ist.

Beispiel:

$$\begin{aligned} plus(\mathcal{O}, y) &\rightarrow y \\ plus(succ(x), y) &\rightarrow succ(plus(x, y)) \end{aligned}$$

$$\begin{aligned} plus(\mathcal{O}, y) &\succ_{emb} y \\ plus(succ(x), y) &\not\succ_{emb} plus(x, y) \end{aligned}$$

4.4 Simplifikationsordung und rekursive Pfadordnung

Das Ziel dieses Abschnitts ist es, eine "stärkere" Reduktionsordnung als $\succ_{emb}$ zu entwickeln, d. h. eine Reduktionsordnung, die $\succ_{emb}$ enthält.

Definition 4.4 (Simplifikationsordnung) *Eine Relation $\succ$ mit $s \succ t$ für alle $s \succ_{emb} t$ heißt **Simplifikationsordnung**.*

Man kann zeigen, dass man in obiger Definition statt der *Fundiertheit* nur die *Irreflexivität* fordern muss. Dies vereinfacht die Überprüfung, ob es sich bei einer Relation um eine Simplifikationsordnung handelt.

Satz 4.6 (Kruskal, 1960) *Es gilt:*

(1) Für jede unendliche Folge von Grundtermen $t_0, t_1, \ldots$ existieren $i, j \in \mathbb{N}$ mit $i < j$ und $t_i \prec_{emb} t_j$.

(2) Sei $\succ$ eine Relation, die stabil, monoton, transitiv ist und die Teiltermeigenschaft erfüllt ($f(x_1, \ldots, x_n) \succ x_i$ für alle i und alle $f \in \Sigma$), dann gilt auch $\succ_{emb} \subseteq \succ$.

(3) Sei $\succ$ eine Relation, die stabil, monoton, transitiv, irreflexiv ist und die Teiltermeigenschaft erfüllt. Dann ist $\succ$ fundiert und daher eine Simplifikationsordnung sowie eine Reduktionsordnung.

Beweis:

(3) Annahme: $\succ$ sei nicht fundiert
$\curvearrowright$ es existiert $t_0 \succ t_1 \succ t_2 \succ \ldots$
Sei σ eine Substitution, die alle Variablen von $t_0, t_1, t_2, \ldots$ durch Grundterme ersetzt. Dies sind nur endlich viele Variablen, denn $\mathcal{V}(t_0) \supseteq \mathcal{V}(t_1) \supseteq \mathcal{V}(t_2) \supseteq \ldots$.
Wegen der Stabilität von $\succ$ ist $t_0\sigma \succ t_1\sigma \succ t_2\sigma \succ \ldots$ eine unendliche Folge von Grundtermen.
$\overset{(1)}{\curvearrowright}$ $\exists i < j$ mit $t_i\sigma \preceq_{emb} t_j\sigma$
$\overset{(2)}{\curvearrowright}$ $t_i\sigma \preceq t_j\sigma$ und $t_i\sigma \succ \ldots \succ t_j\sigma \succeq t_i\sigma$
$\overset{transitiv}{\curvearrowright}$ $t_i\sigma \succ t_i\sigma \rightsquigarrow$ Widerspruch zur Irreflexivität

q.e.d.

Will man zwei Terme vergleichen, die mit dem selben Funktionssymbol anfangen, so kann man den Vergleich auf die Argumente reduzieren. D. h. man vergleicht Tupel von Termen. Dies kann *lexikographisch* oder als *Multimenge* geschehen. So bekommt man ein generelles Konstruktionsprinzip, um aus einfachen Reduktionsordungen komplexere zu konstruieren.

Definition 4.5 (lexikographische Kombination von Relationen) *Seien $\succ_1 \subseteq T_1 \times T_1$ und $\succ_2 \subseteq T_2 \times T_2$. Wir definieren die **lexikographische Kombination** $\succ_{1\times 2} \subseteq T_1 \times T_2$ wie folgt:*

$$(s_1, s_2) \succ_{1\times 2} (t_1, t_2) :\curvearrowright\curvearrowright s_1 \succ_1 t_1 \text{ oder } (s_1 = t_1 \text{ und } s_2 \succ_2 t_2)$$

Analog dazu kann man $\succ_1, \ldots, \succ_n$ lexikographisch kombinieren:

$$(s_1, \ldots, s_n) \succ_{1\times\ldots\times n} (t_1, \ldots, t_n) \quad :\curvearrowright\curvearrowright \quad \exists i \in \{1, \ldots, n\} \text{ mit } s_i \succ_i t_i \text{ und } \forall j \in \{1, \ldots, i-1\} : s_j = t_j$$

Die n-fache Kombination einer Relation $\succ$ mit sich selbst wird mit $\succ^n_{lex}$ bezeichnet.

Beispiel:

- $(3,5)(>_{\mathbb{N}})^2_{lex}(3,2)$
- $(3,5)(>_{\mathbb{N}})^2_{lex}(2,10)$
- $a >_{alph} b >_{alph} \ldots >_{alph} z$
  $hans(>_{alph})^4_{lex} hugo$

Beachte:

Wichtig ist, dass die Größe der Tupel, die verglichen werden, gleich bleibt. Sonst wäre die *lexikographische Kombination* nicht fundiert.

Satz 4.7 *Seien $\succ_1 \subseteq T_1 \times T_1$, $\succ_2 \subseteq T_2 \times T_2$ und $T_1 \neq \emptyset \neq T_2$. $\succ_1$ und $\succ_2$ sind fundiert $\curvearrowleft\curvearrowright$ $\succ_{1\times 2}$ ist fundiert.*

Beweis:

"$\curvearrowleft$" Sei $\succ_{1\times 2}$ fundiert.
Falls $\succ_1$ nicht fundiert ist, existiert $u_0 \succ_1 u_1 \succ_1 u_2 \succ_1 \ldots$
Sei $v \in T_2$ beliebig: $(u_0, v) \succ_{1\times 2} (u_1, v) \succ_{1\times 2} \ldots$
$\rightsquigarrow$ Widerspruch zur Voraussetzung
Falls $\succ_2$ nicht fundiert ist, existiert $v_0 \succ_2 v_1 \succ_2 v_2 \succ_2 \ldots$
Sei $u \in T_1$ beliebig: $(u, v_0) \succ_{1\times 2} (u, v_1) \succ_{1\times 2} \ldots$
$\rightsquigarrow$ Widerspruch zur Voraussetzung

"$\curvearrowright$" Annahme: $(u_0, v_0) \succ_{1\times 2} (u_1, v_1) \succ_{1\times 2} \ldots$
$\curvearrowright u_0 \succeq_1 u_1 \succeq_1 u_2 \succeq_1 \ldots$
Weil $\succ_1$ fundiert ist, existiert ein $i \in \mathbb{N}$ mit $u_i = u_{i+1} = \ldots$
$\curvearrowright v_i \succ_2 v_{i+1} \succ_2 v_{i+2} \succ_2 \ldots$
$\rightsquigarrow$ Widerspruch zur Fundiertheit von $\succ_2$

q.e.d.

Definition 4.6 (lexikographische Pfadordnung) *Für eine Sigantur Σ und eine Ordnung ("Präzedenz") $\sqsupset$ über Σ ist **lexikographische Pfadordnung** $\succ_{lpo}$ über $\mathcal{T}(\Sigma, \mathcal{V})$ definiert als $s \succ_{lpo} t :\curvearrowleft\curvearrowright$*

(1) $s = f(s_1, \ldots, s_n)$ und $s_i \succeq_{lpo} t$ für ein $i \in \{1, \ldots, n\}$ oder

(2) $s = f(s_1, \ldots, s_n)$, $t = g(t_1, \ldots, t_m)$, $f \sqsupset g$ und $s \succ_{lpo} t_j$ für alle $1 \leq j \leq m$ oder

(3) $s = f(s_1, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots, s_n)$, $t = f(s_1, \ldots, s_{i-1}, t_i, t_{i+1}, \ldots, t_n)$, $s_i \succ_{lpo} t_i$ und $s \succ_{lpo} t_j$ für alle $i < j \leq n$

Beispiel:

$plus(succ(x), y) \succ_{lpo} succ(plus(x, y))$
Um Terme mit unterschiedlichen äußeren Funktionssymbolen – hier: *plus* und *succ* – zu vergleichen, verwendet man eine Ordnung $\sqsupset$ auf den Funktionssymbolen – hier: $plus \sqsupset succ$.

$$
\begin{array}{rcl}
sum(\mathcal{O}, y) & \rightarrow & y \\
sum(\mathcal{O}, y) & \succ_{lpo} & y
\end{array}
$$

$$
\begin{array}{rcl}
sum(succ(x), y) & \rightarrow & sum(x, succ(y)) \\
succ(x) \succ_{lpo} x & \text{und} & sum(succ(x), y) \succ_{lpo} succ(y) \\
sum(succ(x), y) & \succ_{lpo} & sum(x, succ(y))
\end{array}
$$

$$
\begin{array}{rcl}
sum & \sqsupset & succ
\end{array}
$$

Satz 4.8 *Falls $\sqsupset$ eine fundierte Ordnung ist, dann ist die zugehörige lexikographische Pfadordnung $\succ_{lpo}$ eine Simplifikationsordnung.*

Beweis:

Wegen Satz 4.6 reicht es zu zeigen, das $\succ_{lpo}$ die Teiltermeigenschaft hat, sowie stabil, monoton, transitiv und irreflexiv ist.

- Teiltermeigenschaft: $f(x_1, \ldots, x_i, \ldots, x_n) \succ_{lpo} x_i$, denn $x_i \succeq_{lpo} x_i$
- Stabilität: z.z.: $s \succ_{lpo} t \curvearrowright s\sigma \succ_{lpo} t\sigma$ für $\sigma \in SUB(\Sigma, \mathcal{V})$
  Verwende NOETHERsche Induktion über $\rhd_{lex}$. Wenn wir die Aussage für s und t zeigen wollen, dürfen wir sie also für alle s' und t' voraussetzen mit $(s, t) \rhd^2_{lex} (s', t')$.
  (1) Sei $s = f(s_1, \ldots, s_n)$ und $s_i \succeq_{lpo} t \overset{I.V.}{\curvearrowright} s_i\sigma \succeq_{lpo} t\sigma$
  Daher gilt: $s\sigma = f(s_1\sigma, \ldots, s_i\sigma, \ldots, s_n\sigma) \succ_{lpo} t\sigma$
  (2) Sei $s = f(s_1, \ldots, s_n)$, $t = g(t_1, \ldots, t_n)$, $f \sqsupset g$ und $s \succ_{lpo} t_j$ für alle $1 \leq j \leq m \overset{I.V.}{\curvearrowright} s\sigma \succ_{lpo} t_j\sigma$
  Also: $s\sigma = f(s_1\sigma, \ldots, s_n\sigma)$, $t\sigma = g(t_1\sigma, \ldots, t_m\sigma)$, $f \sqsupset g$, $\forall j : s\sigma \succ_{lpo} t_j\sigma \curvearrowright s_\sigma \succ_{lpo} t\sigma$
  (3) Sei $s = f(s_1, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots, s_n)$, $t = f(s_1, \ldots, s_{i-1}, t_i, t_{i+1}, \ldots, t_n)$, $s_i \succ_{lpo} t_i$ und $s \succ_{lpo} t_j$ für alle $i < j \leq m$
  $\overset{I.V.}{\curvearrowright} s_i\sigma \succ_{lpo} t_i\sigma$, $s\sigma \succ_{lpo} t_j\sigma$
  Also: $s\sigma = f(s_1\sigma, \ldots, s_i\sigma, \ldots, s_n\sigma)$, $t\sigma = f(s_1\sigma, \ldots, t_i\sigma, \ldots, t_n\sigma) \curvearrowright s\sigma \succ_{lpo} t\sigma$
- Der Rest wird analog mittels NOETHERsche Induktion bewiesen.

q.e.d.

Mit den oben beschriebenen Hilfsmitteln können wir nun einen automatischen Terminierungsbeweis formulieren:

(1) Finde eine fundierte Präzedenz $\sqsupset$, so dass $l \succ_{lpo} r$ für alle Regeln $l \rightarrow r$ gilt.

(2) Lasse dazu $\sqsupset$ zunächst unbestimmt und erweitere sie "*on demand*" Schritt für Schritt, so dass die Fundiertheit nicht verletzt wird.

(3) Überprüfe, ob es zu einem Termersetzungssystem eine solche Präzedenz und eine lexikographische Pfadordnung gibt, die seine Terminierung beweisen kann (entscheidbar)

Beachte:

Es existieren Termersetzungssysteme, deren Terminierung nicht mit einer lexikographischen Pfadordnung gezeigt werden kann.

Beispiel:

$$\begin{aligned} pred(\mathcal{O}) &\rightarrow \mathcal{O} \\ pred(succ(x)) &\rightarrow x \\ minus(x, \mathcal{O}) &\rightarrow x \\ minus(x, succ(y)) &\rightarrow minus(pred(x), y) \end{aligned}$$

$minus(x, succ(y)) \succ_{lpo} minus(pred(x), y)$ gilt nur falls $x \succ_{lpo} pred(x)$, was aber nicht gilt.

Man kann diesem Problem teilweise Abhilfe verschaffen, indem man eine Variante der lexikographischen Pfadordnung entwickelt, bei der die Argumente von rechts nach links (statt von links nach rechts) verglichen werden. D. h. dass die dritte Bedingung in der Definition geändert werden müsste. Analog dazu sind beliebige Varianten der lexikographischen Pfadordnung möglich, bei denen die Argumente in beliebiger Permutation lexikographisch verglichen werden. So erhält man die **lexikographische Pfadordnung mit Status** $\succ_{lpos}$:

Jedes n-stellige Funktionssysmbol erhält einen *Status*, d. h. eine Permutation der Zahlen $1, \ldots, n$, der angibt, in welcher Reihenfolge die Argumente dieses Funktionssymbols verglichen werden sollen.

In obigem Beispiel würde *minus* den Status $\langle 2, 1\rangle$ erhalten, und *plus* $\langle 1, 2\rangle$.

Die *lexikographische Pfadordnung mit Status* bleibt eine *Simplifikationsordung*, und die Frage, ob die Terminierung eines Termersetzungssystems mit Hilfe von $\succ_{lpo}$ mit Status nachgewiesen kann, ist immer noch entscheidbar. Dabei existieren aber auch noch Termersetzungssysteme, deren Terminierung auch nicht mittels $\succ_{lpo}$ mit Status gezeigt werden kann.

Beispiel:

$$\begin{aligned} plus(\mathcal{O}, y) &\rightarrow y \\ plus(succ(x), y) &\rightarrow succ(plus(y, x)) \end{aligned}$$

$plus((succ(x), y) \succ_{lpos} succ(plus(y, x))$, falls

- $plus \sqsupset succ$
- $plus((succ(x), y) \succ_{lpos} plus(y, x)$

$\langle 1, 2\rangle$: $succ(x) \not\succ_{lpos} y$
$\langle 2, 1\rangle$: $y \not\succ_{lpos} x$

$\rightsquigarrow \succ_{lpos}$ scheitert

Die Idee diesem noch bestehendem Problem auch Abhilfe zu verschaffen besteht darin, den Vergleich der Argumente nicht von der Reihenfolge abhängig zu machen, also eine Simplifikationsordnung zu entwerfen, bei der die Argumente nicht als *Tupel* sondern als *Multimengen* verglichen werden. Dazu führen wir zunächst den Begriff der Multimenge ein.

Definition 4.7 (Multimenge) *Sei T eine nicht-leere Menge. Eine ungeordnete endliche Folge von Elementen aus T heißt* ***Multimenge*** *über T. $\mathcal{M}(T)$ ist die* ***Menge aller endlichen Multimengen*** *über T. Eine Multimenge M ist durch ihre* ***charakteristische Funktion***

$$\#_M : T \longrightarrow \mathbb{N}$$

definiert, wobei $\#_M(t)$ angibt, wie oft das Element t in M vorkommt:

- $t \in M \leftrightsquigarrow \#_M(t) > 0$
- $M = N \leftrightsquigarrow \#_M(t) = \#_N(t)$ *für alle* $t \in T$
- $M \subseteq N \leftrightsquigarrow \#_M(t) \leq \#_N(t)$ *für alle* $t \in T$
- $M \cup N$ *ist die Multimenge mit* $\#_{M \cup N}(t) = \#_M(t) + \#_N(t)$ *für alle* $t \in T$
- $M \cap N$ *ist die Multimenge mit* $\#_{M \cap N}(t) = min\{\#_M(t), \#_N(t)\}$ *für alle* $t \in T$
- $M \setminus N$ *ist die Multimenge mit* $\#_{M \setminus N}(t) = max\{0, \#_M(t) - \#_N(t)\}$ *für alle* $t \in T$

Beispiel:

- $M = \{1, 1, 4\}$, $N = \{1, 2, 2, 4, 4, 5\}$
- $M \not\subseteq N$, $N \not\subseteq M$
- $M \cup N = \{1, 1, 1, 2, 2, 4, 4, 4, 5\}$
- $M \cap N = \{1, 4\}$
- $M \setminus N = \{1\}$
- $N \setminus M = \{2, 2, 4, 5\}$

Jede Relation auf einer Menge T kann zu einer Relation auf $\mathcal{M}(T)$ erweitert werden.

Definition 4.8 (Multimengen-Relation) *Sei $\succ \subseteq T \times T$. Dann ist die* ***Multimengen-Relation*** *$\succ_{mul} \subseteq \mathcal{M}(T) \times \mathcal{M}(T)$ definiert als $M \succ_{mul} N :\leftrightsquigarrow$*

- $\emptyset \neq X \subseteq M$
- $N = (M \setminus X) \cup Y$
- $\forall y \in Y : \exists x \in X : x \succ y$

Beispiel:

$M = \{3, 6, 8\}$, $N = \{4, 5, 6, 6, 7, 7\}$
$M (>_{\mathbb{N}})_{mul} N$, denn $X = \{3, 8\}$, $Y = \{4, 5, 6, 7, 7\}$
und $8 >_{\mathbb{N}} 4, 5, 6, 7, 7$

Satz 4.9 *Sei $\succ \subseteq T \times T$. $\succ$ ist fundiert $\leftrightsquigarrow \succ_{mul}$ ist fundiert.*

Beweis:

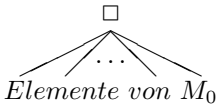
"$\curvearrowleft$" Annahme: $t_0 \succ t_1 \succ \ldots$
$\curvearrowright \{t_0\} \succ_{mul} \{t_1\} \succ_{mul} \ldots$
$\rightsquigarrow$ Widerspruch zur Voraussetzung

"$\curvearrowright$" Annahme: $M_0 \succ_{mul} M_1 \succ_{mul} \ldots$
Konstruktion eines unendlichen Baumes, dessen Knoten mit Elementen aus T markiert sind. Die Elemente nehmen entlang eines Pfades ab (Knoten s hat Kind $t \curvearrowleft\!\!\curvearrowright s \succ t$) und der Baum hat endlichen Verzweigungsgrad. Wegen Satz 4.2 existiert dann also ein unendlicher Pfad im Baum, der einer unendlich absteigenden $\succ$-Folge entspricht, was ein Widerpruch zur Voraussetzung wäre.

Erzeugen des Baumes:
Erweitere T um zusätzliches Element $\bot$, d. h. $T_\bot := T \cup \{\bot\}$. Markiere die Knoten des Baumes mit Elementen aus $T_\bot$. Erweitere $\succ$ auf $T_\bot$ durch $t \succ \bot$ für alle $t \in T$. Die Fundiertheit von $\succ$ bleibt dadurch erhalten.

Sukzessive Konstruktion: Im i-ten Schritt sind die Blätter des Baumes mit M_i und $\bot$ markiert.

0-*ter Schritt:*

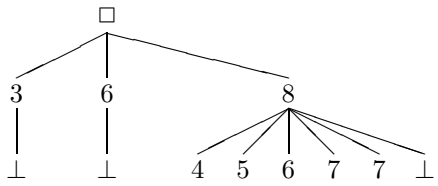


1-*ter Schritt:*
Da $M_0 \succ_{mul} M_1$ gilt: $\exists X \subseteq M_0, Y \subseteq M_1$ wie in der Definition gefordert. Für jedes $y \in Y$ füge neues Blatt mit der Markierung y hinzu und mache daraus ein Kind eines $x \in X$ mit $x \succ y$. Außerdem erhält jeder Knoten $x \in X$ ein weiteres Blatt mit der Markierung $\bot$. Dieser Erweiterungsschritt wird unendlich oft durchgeführt (für M_1, M_2, ...). D. h. es existiert ein unendlicher Pfad, der einer unendlich absteigenden $\succ$-Folge entspricht. $\rightsquigarrow$ Widerpsruch

q.e.d.

Beispiel:

$M_0 = \{3, 6, 8\}$
$M_1 = \{4, 5, 6, 6, 7, 7\}$
$X = \{3, 8\}$
$Y = \{4, 5, 6, 7, 7\}$



Definition 4.9 (Rekursive Pfadordnung) *Für eine Signatur Σ und eine Ordnung $\sqsupset$ auf Σ ist die* ***rekursive Pfadordnung*** $\succ_{rpo}$ *über $\mathcal{T}(\Sigma, \mathcal{V})$ definiert als $s \succ_{rpo} t :\curvearrowleft\!\!\curvearrowright$*

(1) $s = f(s_1, \ldots, s_n)$ und $s_i \succeq_{rpo} t$ für ein $i \in \{1, \ldots, n\}$ oder

(2) $s = f(s_1, \ldots, s_n)$, $t = g(t_1, \ldots, t_m)$, $f \sqsupset g$ *und* $s \succ_{rpo} t_j$ *für alle* $1 \leq j \leq m$ *oder*

(3) $s = f(s_1, \ldots, s_n)$, $t = f(t_1, \ldots, t_n)$ *und* $\{s_1, \ldots, s_n\}(\succ_{rpo})_{mul}\{t_1, \ldots, t_n\}$

Beispiel:

Die lexikographische Pfadordnung scheitert am *plus*-TES mit vertauschter Reihenfolge der Argumente bei der 2. Regel. Die rekursive Pfadordung hingegen kann die Terminierung dieses TES beweisen:

$$\begin{array}{rcl} plus(succ(x), y) & \succ_{rpo} & succ(plus(y, x)) \\ plus & \sqsupset & succ \end{array}$$

$$\begin{array}{rcl} plus(succ(x), y) & \succ_{rpo} & plus(y, x) \\ \{succ(x), y\} & (\succ_{rpo})_{mul} & \{y, x\} \end{array}$$

$$\begin{array}{rcl} succ(x) & \succ_{rpo} & x \\ x & \succeq_{rpo} & x \end{array}$$

Vorteil der rekursiven Pfadordung:

Das Vertauschen von Argumenten ist unproblematisch.

Nachteil der rekursiven Pfadordung:

Argumente dürfen nicht "größer" werden.

Satz 4.10 *Falls* $\sqsupset$ *eine fundierte Ordunug ist, dann ist* $\succ_{rpo}$ *eine Simplifikationsordung.*

Beweis:

Analog zum Beweis von Satz 4.8.

q.e.d.

Es gibt also Termersetzungssysteme, deren Termination mit Hilfe der lexikographischen nicht aber der rekursiven Pfadordnung gezeigt werden kann und umgekehrt. Daher ist es sinnvoll diese beiden Pfadordungen zu verbinden. Dies kann man erreichen, indem man jedem n-stelligem Funktionssymbol einen *Status* zuweist; ein Status ist entweder eine Permutation von $1, \ldots, n$ oder "Multimenge". Er gibt an, auf welche Weise zwei Terme mit gleichen Funktionssymbolen an äußerter Stelle verglichen werden sollen. Somit erhält man die sogenannte **rekursive Pfadordnung mit Status** $\succ_{rpos}$.

Mit deren Hilfe kann man ein *Terminierungsverfahren* formulieren: Suche nach einer Präzedenz mit Status, so dass $l \succ_{rpos} r$ für alle Regeln $l \to r \in \mathcal{R}$ gilt. Dies ist *entscheidbar* und somit haben wir ein automatisches Terminierungsverfahren gefunden.

Leider existieren aber immer noch Termersetzungssysteme, deren Terminierung *nicht* mit Hilfe von $\succ_{rpos}$ oder irgendeiner anderen *Simplifikationsordnung* gezeigt werden kann. D. h. es gibt noch weitere Verbesserungsmöglichkeiten für Terminierungsverfahren.

Kapitel 5

Konfluenz von Termesetzungssystemen

Geht man davon aus, dass ein Termersetzungssystem terminiert, dann ist dessen Konfluenz automatisch entscheidbar.

5.1 Unifikation

Die *Unifikation* wird zur Konfluenzuntersuchung und Vervollständigung von TES benötigt.

Definition 5.1 (Unifikation) *Wir definieren:*

- *Zwei Terme s und t sind **unifizierbar** $:\curvearrowright\curvearrowright$ es existiert eine Substitution σ mit $s\sigma = t\sigma$. σ heißt dann **Unifikator** von s und t.*
- *Ein **Unifikationsproblem** über Σ und $\mathcal{V}$ ist eine endliche Menge von Termgleichungen $S = \{s_1 =^? t_1, \ldots, s_n =^? t_n\}$ mit $s_i, t_i \in \mathcal{T}(\Sigma, \mathcal{V})$.*
- *Eine Substitution σ ist **Lösung** oder **Unifikator** von $S :\curvearrowright\curvearrowright s_i\sigma = t_i\sigma$ für alle $i \in \{1, \ldots, n\}$.*
- *$U(S)$ ist die **Menge aller Unifikatoren** von S.*

Beispiel:

$s = g(f(x), y)$ und $t = g(y, f(z))$

- $\sigma = \{y/f(z), x/z\}$ ist ein Unifikator von s und t
- $\sigma_1 = \{x/a, y/f(a), z/a\}$ ist ein Unifikator von s und t
- $\sigma_2 = \{x/f(z), y/f(f(z)), z/f(z)\}$ ist ein Unifikator von s und t
- $\sigma_3 = \{y/f(z), z/x\}$ ist ein Unifikator von s und t

σ ist *allgemeiner* als σ_1 und σ_2

Definition 5.2 (allgemeinere Substitution) *Eine Substitution σ ist **allgemeiner** als eine Substitution σ', falls es eine Substitution δ gibt mit $\sigma' = \sigma\delta$. σ' heißt dann **spezieller** als σ.*

Beispiel:

- $\sigma_1 = \sigma \circ \{z/a\}$
- $\sigma_2 = \sigma \circ \{z/f(z)\}$
- $\sigma_3 = \sigma \circ \{z/x\}$

Lemma 5.1 *Wenn σ allgemeiner als σ' und σ' allgemeiner als σ ist, dann existiert eine Variablenumbenennung δ mit $\sigma' = \sigma\delta$. Hierbei heißt δ* ***Variablenumbennung****, falls δ injektiv ist und $\{x\delta \mid x \in DOM(\delta)\} \subseteq \mathcal{V}$ gilt.*

Beweis:

[Übung]

q.e.d.

Beispiel:

- $\sigma_3\delta = \sigma$
- $\sigma\delta = \sigma_3$

Es genügt also einen *allgemeinsten Unifikator* (*most general unifier*, MGU) zu berechnen. Die oft unendliche Menge $U(S)$ lässt sich dann durch *einen* Repräsentanten darstellen.

Lemma 5.2 *Sei S ein Unifikationsproblem über Σ und $\mathcal{V}$ und sei σ MGU von S. Dann gilt:*

$$U(S) = \{\sigma\delta \mid \delta \in SUB(\Sigma, \mathcal{V})\}$$

Beweis:

"$\supseteq$": z.z.: $\sigma\delta$ ist Unifikator.
Für alle $s =^? t \in S$ gilt: $s\sigma = t\sigma \curvearrowright s\sigma\delta = t\sigma\delta$

"$\subseteq$": Sei $\sigma' \in U(S)$.
Dann existiert δ mit $\sigma' = \sigma\delta$, da σ allgemeinster Unifikator

q.e.d.

Das weitere Ziel dieses Abschnitts ist es, einen Algorithmus zu entwickeln, der ein Unifikationsproblem solange transformiert, bis es in *gelöster Form* ist und man den MGU direkt ablesen kann.

Definition 5.3 (gelöste Form eines Unifikationsproblems) *Ein Unifikationsproblem $S = \{x_1 =^? t_1, \ldots, x_n =^? t_n\}$ ist in* ***gelöster Form*** *:$\curvearrowright\curvearrowright$ die x_i sind paarweise verschiedene Variablen, die nicht in den t_j vorkommen (occur check). Wir definieren die Substitution $\sigma_S := \{x_1/t_1, \ldots, x_n/t_n\}$.*

Lemma 5.3 *Sei S ein Unifikationsproblem in gelöster Form. Dann ist σ_S ein MGU von S, wobei für alle $\sigma' \in U(S)$ gilt: $\sigma' = \sigma_S\sigma'$*

Beweis:

(1) $\sigma_S \in U(S)$, denn $x_i\sigma_S \overset{Def}{=} t_i \overset{Def}{=} t_i\sigma_S$.

(2) σ_S ist MGU, denn:
Sei $\sigma' \in U(S)$, z.z.: $\sigma' = \sigma_S\sigma'$
Zeige: für alle $x \in \mathcal{V} : x\sigma' = x\sigma_S\sigma'$

(a) $x \in \{x_1 \ldots, x_n\}$
$\curvearrowright x\sigma' = t_i\sigma' = \underbrace{x_i\sigma_S}_{t_i}\sigma'$

(b) $x \notin \{x_1 \ldots, x_n\}$
$\curvearrowright x\sigma' = x_i\sigma_S\sigma'$, da $x \notin DOM(\sigma_S)$

q.e.d.

Definition 5.4 (Transformation von Unifikationsproblemen) *Die Relation* $\Longrightarrow$ *auf Unifikationsproblemen wird durch 4 Transformationsregeln definiert:*

- ***Löschen**:*
  $S \uplus \{t =^? t\} \Longrightarrow S$
- ***Termreduktion**:*
  $S \uplus \{f(s_1, \ldots, s_n) =^? f(t_1, \ldots, t_n)\} \Longrightarrow S \uplus \{s_1 =^? t_1, \ldots s_n =^? t_n\}$
- ***Vertauschen**:*
  $S \uplus \{t =^? x\} \Longrightarrow S \cup \{x =^? t\}$, *falls* $t \notin \mathcal{V}$
- ***Variablenreduktion**:*
  $S \uplus \{x =^? t\} \Longrightarrow \{u\sigma =^? v\sigma \mid u =^? v \in S\} \cup \{x =^? t\}$, *falls* $x \notin \mathcal{V}(t)$, $x \in \mathcal{V}(S)$, $\sigma = \{x/t\}$

Beispiel:

$\{g(f(x), y) =^? g(y, f(z))\}$ $\Longrightarrow_{Termreduktion}$
$\{f(x) =^? y, y =^? f(z)\}$ $\Longrightarrow_{Vertauschen}$
$\{y =^? f(x), y =^? f(z)\}$ $\Longrightarrow_{Variablenreduktion}$
$\{f(z) =^? f(x), y =^? f(z)\}$ $\Longrightarrow_{Termreduktion}$
$\{z =^? x, y =^? f(z)\}$ $\Longrightarrow_{Variablenreduktion}$
$\{z =^? x, y =^? f(x)\}$
$\curvearrowright \sigma = \{z/x, y/f(x)\}$

Algorithmus:
UNIFY(S)

Eingabe: Unifikationsproblem S

Ausgabe: MGU(S), falls S lösbar, sonst "false"

Vorgehen:

(1) Solange es S' mit $S \Longrightarrow S'$ gibt, setze $S' := S$ und gehe zu (1)

(2) Falls S in gelöster Form, gibt σ_S aus, sonst "false"

Satz 5.1 (Korrekheit von UNIFY) *Es gilt:*

(1) $\Longrightarrow$ *ist fundiert.*

(2) $S \Longrightarrow S' \curvearrowright U(S) = U(S')$

(3) Falls S *lösbar und in Normalform bezüglich* $\Longrightarrow$ *ist, so ist* S *ist gelöster Form*

(4) UNIFY *terminiert und ist korrekt*

Beweis:

(1) Bilde jedes Unifikationsproblem auf $(n_1, n_2, n_3) \in \mathbb{N}^3$ ab mit

- n_1: Anzahl der Variablen in S, die noch nicht gelöst sind
- n_2: Anzahl der Zeichen in S
- n_3: Anzahl der Gleichungen $t =^? x$ in S mit $t \notin \mathcal{V}$

Wie verändern sich n_1, n_2, n_3 bei den Transformationsregeln?

	n_1	n_2	n_3
Löschen	$\geq$	$>$	
Termreduktion	$\geq$	$>$	
Vertauschen	$\geq$	$=$	$>$
Variablenreduktion	$>$		

$S \mapsto (n_1, n_2, n_3)$, $S \Longrightarrow S'$, $S' \mapsto (n_1', n_2', n_3')$
$\curvearrowright (n_1, n_2, n_3)(>_{\mathbb{N}})^3_{lex}(n_1, n_2, n_3)$
$(>_{\mathbb{N}})^3_{lex}$ fundiert $\curvearrowright \Longrightarrow$ fundiert

(2) Zeige $U(S) = U(S')$, falls S' aus S mit *Variablenreduktion* entsteht (für die anderen Regeln ist die Korrektheit klar).
Wenn $\{x =^? t\}$ ist gelöster Form, dann gilt für alle Θ mit $\Theta(x) = \Theta(t)$: $\Theta = \sigma\Theta$ $(\sigma = \{x/t\})$

$\Theta \in U(S \uplus \{x =^? t\})$
$\curvearrowright\curvearrowright$ $x\Theta = t\Theta$ und $\Theta \in U(S)$
$\curvearrowright\curvearrowright$ $x\Theta = t\Theta$ und $\sigma\Theta \in U(S)$
$\curvearrowright\curvearrowright$ $x\Theta = t\Theta$ und $u\sigma\Theta = v\sigma\Theta$ für alle $u =^? v \in S$
$\curvearrowright\curvearrowright$ $x\Theta = t\Theta$ und $\sigma\Theta \in U(\{\alpha u\sigma =^? v\sigma \mid u =^? v \in S\})$
$\curvearrowright\curvearrowright$ $x\Theta = t\Theta$ und $\sigma\Theta \in U(\underbrace{\{u\sigma =^? v\sigma \mid u =^? v \in S\} \cup \{x =^? t\}}_{=S'})$

(3) Sei S lösbar und in Normalform bzgl. $\Longrightarrow$. S kann keine Gleichungen der Form

- $f(\ldots) =^? g(\ldots)$ enthalten, da diese Gleichung nicht lösbar ist.
- $f(\ldots) =^? f(\ldots)$ enthalten, da bei dieser Gleichung die *Termreduktion* anwendbar wäre.
- $t =^? x$ $(t \notin \mathcal{V})$ enthalten, da bei dieser Gleichung *Vertauschen* anwendbar wäre.

$\curvearrowright S = \{x_1 =^? t_1, \ldots, x_n =^? t_n\}$. Hierbei gilt $x_i \notin \mathcal{V}(t_i)$, da sonst *Löschen* anwendbar wäre. Außerdem kommen die paarweise verschiedenen x_i auch in keinem t_j vor, da sonst *Variablenreduktion* anwendbar wäre.
$\curvearrowright S$ ist in gelöster Form.

(4) Die *Terminierung* von UNIFY folgt aus (1).
$\curvearrowright$ UNIFY berechnet die Normalform S' von S, d. h. $S \Longrightarrow^* S'$.
Aus (2) folgt durch Induktion über die Länge der Transformationen: $U(S) = U(S')$.

(a) S lösbar, also $U(S) \neq \emptyset$
$\curvearrowright S'$ lösbar $\curvearrowright S'$ in gelöster Form
$\overset{L5.3}{\curvearrowright}$ $\sigma_{S'}$ ist MGU von S' und damit von S

(b) S nicht lösbar, also $U(S) = \emptyset$
$\curvearrowright U(S') = \emptyset$ $\curvearrowright S'$ nicht in gelöster Form
$\curvearrowright$ UNIFY gibt "false" aus.

q.e.d.

Korollar 5.1 (Unifikationssatz) *Wenn ein Unifikationsproblem lösbar ist, dann besitzt es auch einen allgemeinsten Unifikator.*

Der Aufwand von UNIFY ist exponentiell, lässt sich aber zu einen Algorithmus mit linearem Aufwand verbessern. Ferner lässt sich aus UNIFY ein *Matching*-Algorithmus gewinnen:

Algorithmus:
MATCHER(s, t)

Eingabe: 2 Terme s und t

Ausgabe: *Matcher* σ von s und t, falls einer existiert, sonst "false"

Vorgehen:

(1) Sei $\Theta = \{x/c_x \mid x \in \mathcal{V}(t)\}$, wobei c_x neue Konstanten

(2) Berechne UNIFIY$(\{s =^? t\Theta\})$

(3) Falls UNIFIY "false" liefert, gib ebenfalls "false" aus und brich ab.

(4) Falls UNIFIY $\{x_1/t_1, \ldots, x_n/t_n\}$ liefert, gib $\{x_1/t'_1, \ldots, x_n/t'_n\}$ aus, wobei t'_i aus t_i entsteht, indem die neuen Konstanten c_x durch x ersetzt werden.

Beispiel:

Matching von $g(x, y)$ und $g(f(x), x)$
$\rightsquigarrow$ MATCH$(g(x, y), g(f(x), x))$
$\rightsquigarrow$ UNIFY$(\{g(x, y) =^? g(f(c_x), c_x)\})$
$\rightsquigarrow \{x/f(c_x), y/c_x\}$
$\rightsquigarrow \{x/f(x), y/x\}$

5.2 Lokale Konfluenz und kritische Paare

Die Konfluenz ist im Allgemeinen unentscheidbar; sie ist aber für terminierende Termersetzungssysteme entscheidbar. Dazu schwächen wir zunächst die Konfluenzforderung etwas ab, und verlangen nicht mehr, dass Indeterminismen in *beliebig vielen* Schritten, sondern nur noch nach *einem* Schritt zusammenführbar sein sollen.

Definition 5.5 (lokale Konfluenz) *Eine Relation $\rightarrow \subseteq M \times M$ heißt* ***lokal konfluent*** $:\curvearrowleft\!\curvearrowright \forall s, t, p \in M : p \rightarrow s$ *und* $p \rightarrow t \curvearrowright \exists q \in M : s \rightarrow^* q$ *und* $t \rightarrow^* q$. *Ein Termersetzungssystem $\mathcal{R}$ heißt* ***lokal konfluent*** $:\curvearrowleft\!\curvearrowright \rightarrow_{\mathcal{R}}$ *ist lokal konfluent.*

Beispiel:

$\mathcal{R} := \{b \rightarrow a, b \rightarrow c, c \rightarrow b, c \rightarrow d\}$

$\mathcal{R}$ ist lokal konfluent: $a \leftarrow b \rightarrow c \rightarrow b \rightarrow a$ und $d \leftarrow c \leftarrow b \leftarrow c \rightarrow d$
aber nicht konfluent: $a \leftarrow^* b \rightarrow^* d$

Satz 5.2 (Diamond Lemma, Satz von NEWMAN, 1942) *Sei $\rightarrow$ eine fundierte Relation. $\rightarrow$ ist lokal konfluent $\curvearrowleft\curvearrowright$ $\rightarrow$ ist konfluent.*

Beweis:

"$\curvearrowleft$" ist klar. Z.z.: lokale Konfluenz $\curvearrowright$ Konfluenz.

Sei $\rightarrow$ fundiert und lokal konfluent.
Zeige nun: $\forall p : s \leftarrow^* p \rightarrow^* t \curvearrowright \exists q : s \rightarrow^* q \leftarrow^* t$

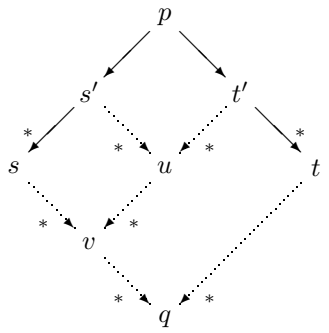
Beweis durch NOETHERsche Induktion über p mit Induktionsrelation $\rightarrow$:

Sei $s \leftarrow^* p \rightarrow^* t$

(1) $p = s$
Dann gilt: $s = p \rightarrow^* t \leftarrow^* t$ ✓

(2) $p = t$
Dann gilt: $t = p \rightarrow^* s \leftarrow^* s$ ✓

(3) $s \neq p \neq t$, also: $s \leftarrow^+ p \rightarrow^+ t$
Dann existieren s', t', u mit: $s' \rightarrow^* u \leftarrow^* t'$ wegen lokaler Konfluenz



Da $p \rightarrow s'$, existiert aufgrund der Induktionsvoraussetzung ein v mit $s \rightarrow^* v \leftarrow^* u$. Da $p \rightarrow t'$, existiert aufgrund der Induktionsvoraussetzung ein q mit $v \rightarrow^* q \leftarrow^* t$.

q.e.d.

Um die Konfluenz von terminierenden Systemen zu entscheiden, kann die Konfluenz *lokalisiert* werden. Man untersucht also die lokale Konfluenz, indem man alle *kritischen Situationen* der Form $s \overset{R_1}{\leftarrow} p \overset{R_2}{\rightarrow} t$ betrachet, wobei $R_i = l_i \rightarrow r_i$ Regeln aus dem zu untersuchendem Termersetzungssystem sind. Man muss daher *endlich viele kritische Situationen* identifizieren, so dass aus der Zusammenführbarkeit dieser die Zusammenführbarkeit *aller Situationen* folgt.

Untersuche, an welchen Stellen die Regeln $R_1 = l_1 \rightarrow r_1$ und $R_2 = l_2 \rightarrow r_2$ auf p angewendet werden können: R_i wurde an Stelle π_i mit der Substitution σ_i angewendet:

$$\begin{aligned} p|_{\pi_i} &= l_i\sigma_i \\ s &= p[r_1\sigma_1]_{\pi_1} \\ t &= p[r_2\sigma_2]_{\pi_2} \end{aligned}$$

Wir werden im Folgenden die verschiedenen Möglichkeiten betrachten, wie sich π_1 und π_2 zueinander verhalten können.

(1) $\pi_1 \perp \pi_2$ (π_1 und π_2 unabhängig voneinander)

$p = p[l_1\sigma_1]_{\pi_1}[l_2\sigma_2]_{\pi_2}$ und
$s = p[r_1\sigma_1]_{\pi_1}[l_2\sigma_2]_{\pi_2}$, $t = p[l_1\sigma_1]_{\pi_1}[r_2\sigma_2]_{\pi_2}$ sowie
$q = p[r_1\sigma_1]_{\pi_1}[r_2\sigma_2]_{\pi_2}$

Solche Situationen sind immer zusammenführbar und müssen daher für die lokale Konfluenz nicht betrachtet werden.

(2) $\pi_1 \geq_{\mathbb{N}^*} \pi_2$ (π_1 liegt über π_2)

Es existiert ein π mit $\pi_2 = \pi_1\pi$. Anstelle von $s \leftarrow p \rightarrow t$ betrachte nur $s|_{\pi_1} \leftarrow p|_{\pi_1} \rightarrow t|_{\pi_1}$.
Wenn $s|_{\pi_1} \downarrow t|_{\pi_1}$, dann gilt nämlich auch $s \downarrow t$

Grund: $p = p[l_1\sigma_1]_{\pi_1} = p[l_1\sigma_1[l_2\sigma_2]_{\pi}]_{\pi_1}$ und
$s = p[r_1\sigma_1]_{\pi_1}$, $t = p[r_2\sigma_2]_{\pi_2} = p[l_1\sigma_1[r_2\sigma_2]_{\pi}]_{\pi_1}$
$p|_{\pi_1} = l_1\sigma_1[l_2\sigma_2]_{\pi}$
$s|_{\pi_1} = r_1\sigma_1$
$t|_{\pi_1} = l_1\sigma_1[r_2\sigma_2]_{\pi}$

Falls $s|_{\pi_1} \rightarrow^* q \leftarrow t|_{\pi_1}$, dann $s \rightarrow^* p[q]_{\pi_1} \leftarrow t$

(a) $\pi \notin Occ(l_1)$ oder $l_1|_\pi \in \mathcal{V}$:
Dieser Fall ist unkritisch, da man die Terme in (möglicherweise mehreren Schritten) zusammenführen kann.

(b) $\pi \in Occ(l_1)$ und $l_1|_\pi \notin \mathcal{V}$:
Kritische Überlappung: $l_1|_\pi\sigma_1 = l_2\sigma_2$
Nenne die Variablen in $l_1 \rightarrow r_1$ und $l_2 \rightarrow r_2$ so um, dass die Regeln keine gemeinsamen Variablen haben.
$l_1|_\pi$ und l_2 sind unifizierbar mit MGU $\sigma := \sigma_1\sigma_2$

Betrachte also nur die folgenden kritischen Situationen mit dem allgemeinsten Unifikator σ:

$$r_1\sigma \leftarrow \underbrace{l_1[l_2]_\pi\sigma}_{=l_1\sigma} \rightarrow l_1[r_2]_\pi\sigma$$

Beispiel:

$$\begin{aligned} f(x, f(y, z)) &\rightarrow f(f(x, y), z) \\ f(x, e) &\rightarrow x \\ f(x, i(x)) &\rightarrow e \end{aligned}$$

Forme die zweite Regel um in $f(x', e) \rightarrow x'$

$l_1 = f(x, f(y, z))$
$\pi = 2$
$l_1|_\pi = f(y, z)$
$l_2 = f(x', e)$
$\sigma = \{x'/y, z/e\}$
$l_1\sigma = l_1[l_2]_\pi\sigma = f(x, f(y, e))$

$\curvearrowright \langle f(f(x, y), e), f(x, y)\rangle$ ist ein *kritisches Paar*

Definition 5.6 (kritisches Paar) *Sei $\mathcal{R}$ ein Termersetzungssystem, $l_1 \to r_1, l_2 \to r_2 \in \mathcal{R}$. Hierbei seien die Variablen so umbenannt, dass $\mathcal{V}(l_1) \cap \mathcal{V}(l_2) = \emptyset$. Sei $\pi \in Occ(l_1)$ mit $l_1|_\pi \notin \mathcal{V}$ und sei σ allgemeinster Unifikator von $l_1|_\pi$ und l_2. Dann ist $\langle r_1\sigma, l_1[r_2]_\pi\sigma\rangle$ ein **kritisches Paar** von $\mathcal{R}$. Die Regeln $l_1 \to r_1$ und $l_2 \to r_2$ können (bis auf Variablenumbenennungen) identisch sein. In diesem Fall betrachtet man nur Überlappungen mit $\pi \neq \varepsilon$. $CP(\mathcal{R})$ ist **die Menge aller kritischen Paare** von $\mathcal{R}$.*

Satz 5.3 (kritisches Paar Lemma) *Sei $\mathcal{R}$ ein Termersetzungssystem. Dann ist $\mathcal{R}$ lokal konfluent $\curvearrowright\!\!\curvearrowleft$ alle kritschen Paare sind zusammenführbar.*

Beweis:

siehe oben

q.e.d.

Algorithmus:
CONFLUENCE($\mathcal{R}$)

Eingabe: terminierendes Termersetzungssystem $\mathcal{R}$

Ausgabe: "true", falls $\mathcal{R}$ konfluent ist, sonst "false"

Vorgehen:

(1) Berechne $CP(\mathcal{R})$ mittels UNIFY

(2) Falls $CP(\mathcal{R}) = \emptyset$, gib "true" aus und brich ab

(3) Wähle $\langle s, t\rangle \in CP(\mathcal{R})$

(4) Reduziere s, t solange wie möglich. So entstehen die Normalformen $s\!\downarrow$ und $t\!\downarrow$

(5) Falls $s\!\downarrow \neq t\!\downarrow$, gib "false" aus und brich ab

(6) $CP(\mathcal{R}) := CP(\mathcal{R}) \setminus \{\langle s, t\rangle\}$

(7) Gehe zu (2)

Satz 5.4 *Es gilt:*

(1) CONFLUENCE *terminiert und ist korrekt.*

(2) Für terminierende Termersetzungssysteme ist die Konfluenz entscheidbar.

Beweis:

(1) Die *Terminierung* folgt

- aus der Terminierung von UNIFY,
- weil $CP(\mathcal{R})$ endlich ist,
- weil die Berechnung der Normalformen terminiert und
- weil $CP(\mathcal{R})$ bei jedem Durchlauf (2)–(7) kleiner wird

Der Algorithmus gibt nur "true" aus, wenn alle kritischen Paare zusammenführbar sind
$\overset{S5.3}{\curvearrowright}$ Termersetzungssystem ist lokal konfluent

$\overset{L5.2}{\curvearrowright}$ Termersetzungssystem ist konfluent
Falls "false" ausgegeben wird existiert $s\downarrow \leftarrow^* s \leftarrow p \rightarrow t \rightarrow^* t\downarrow$
$\curvearrowright$ Termersetzungssystem ist nicht konfluent
$\curvearrowright$ *Korrektheit*

(2) CONFLUENCE ist ein *Entscheidungsalgorithmus.*

q.e.d.

Kapitel 6

Vervollständigung von Termersetzungssystemen

Das Ziel dieses Kapitels ist die automatische Generierung eines Entscheidungsverfahrens für das Wortproblem über einer Gleichungsmenge $\mathcal{E}$.

6.1 Das grundlegende Vervollständigungs-Verfahren

Beispiel:

$\mathcal{E} := \{plus(\mathcal{O}, y) \equiv y, plus(succ(x), y) \equiv succ(plus(x, y))\}$

$\mathcal{R} := \{plus(\mathcal{O}, y) \to y, plus(succ(x), y) \to succ(plus(x, y))\}$

$\mathcal{R}$ terminiert mit $\succ_{lpo}$ und $plus \sqsupset succ$. Da $CP(\mathcal{R}) = \emptyset$ ist $\mathcal{R}$ zudem konfluent.

Beispiel:

Sei $\mathcal{E}$ das Gleichungssystem der Gruppenaxiome und $\mathcal{R}$ das TES der von links nach rechts gerichteten Gleichungen aus $\mathcal{E}$. $\mathcal{R}$ terminiert mit $\succ_{lpo}$ mit Status, wobei $f : \langle 2, 1\rangle$ und $f \sqsupset e$, aber $\mathcal{R}$ ist nicht konfluent.

Die kritschen Paare identifizieren den Grund für die Zerstörung der Konfluenz. Wenn $\langle s, t\rangle \in CP(\mathcal{R})$, $s \to^* s\downarrow$, $t \to^* t\downarrow$ und $s\downarrow \neq t\downarrow$, dann ergänze $\mathcal{R}$ um die neue Regel $s\downarrow \to t\downarrow$ oder $t\downarrow \to s\downarrow$. Wenn die Terminierung von $\mathcal{R}$ mit $\succ$ gezeigt wurde, dann $s' \to t'$, falls $s' \succ t'$ bzw. umgekehrt. So bleibt die Terminierung erhalten.

Da so nur neue Regeln hinzukommen gilt $\leftrightarrow^*_{\mathcal{E}} = \leftrightarrow^*_{\mathcal{R}_{alt}} \subseteq \leftrightarrow^*_{\mathcal{R}_{neu}}$ wegen der Adäquatheit. Neue Regeln haben die Form $s' \to t'$ für $s' \leftarrow p \to t'$, d. h. $s' \leftrightarrow^*_{\mathcal{R}_{alt}} t'$. Daraus folgt, dass alle kritischen Paare nur $\mathcal{E}$-gleiche Terme in Beziehung setzen. Somit gilt:

$$\leftrightarrow^*_{\mathcal{E}} = \leftrightarrow^*_{\mathcal{R}_{alt}} = \leftrightarrow^*_{\mathcal{R}_{neu}}$$

Die Konfluenz muss dabei aber noch überprüft werden, da zwar alle alten kritischen Paare zusammenführbar sind, aber möglicherweise neue kritische Paare entstanden sind, die nicht zusammenführbar sind. Damit können wir den folgenden Algorithmus formulieren.

Algorithmus:
BASIC-COMPLETION$(\mathcal{E}, \succ)$

Eingabe: Gleichungssystem $\mathcal{E}$, Reduktionsordung $\succ$

Ausgabe: Ein zu $\mathcal{E}$ äquivalentes, konvergentes Termersetzungssystem $\mathcal{R}$, "fail" oder Nichtterminierung

Vorgehen:

(1) Falls es $s \equiv t \in \mathcal{E}$ gibt mit $s\downarrow \neq t\downarrow$ sowie $s\downarrow \not\succ t\downarrow$ und $t\downarrow \not\succ s\downarrow$ gib "fail" aus und brich ab.

(2) Setze $i := 0$ und $\mathcal{R}_0 := \{l \to r \mid l \succ r \text{ und } l \equiv r \in \mathcal{E} \vee r \equiv l \in \mathcal{E}\}$

(3) Setze $\mathcal{R}_{i+1} := \mathcal{R}_i$

(4) Für alle $\langle s, t\rangle \in CP(\mathcal{R}_i)$:

  (a) Berechne $\mathcal{R}_i$-Normalform s' und t' von s und t.

  (b) Falls $s\downarrow \neq t\downarrow$ sowie $s\downarrow \not\succ t\downarrow$ und $t\downarrow \not\succ s\downarrow$ gib "fail" aus und brich ab.

  (c) Falls $s' \succ t'$, dann setze $\mathcal{R}_{i+1} := \mathcal{R}_i \cup \{s' \to t'\}$

  (d) Falls $t' \succ s'$, dann setze $\mathcal{R}_{i+1} := \mathcal{R}_i \cup \{t' \to s'\}$

(5) Falls $\mathcal{R}_{i+1} = \mathcal{R}_i$, dann gib "true" aus und brich ab.

(6) Setze $i := i + 1$ und gehe zu (3)

Es gibt also beim Algorithmus BASIC-COMPLETION drei mögliche Ergebnisse:

(1) Erfolg:
Man erreicht ein $\mathcal{R}_n$, dessen kritische Paare zusammenführbar sind. $\mathcal{R}_n$ ist die Ausgabe des Algorithmus'.

(2) Fehlschlag:
Man erreicht ein $\mathcal{R}_n$ mit einem kritischen Paar $\langle s, t\rangle$ und $s\downarrow \neq t\downarrow$ sowie $s\downarrow \not\succ t\downarrow$ und $t\downarrow \not\succ s\downarrow$. Der Algorithmus gibt "fail" aus. Man könnte die Vervollständigung aber mit einer anderen Reduktionsordung probieren.

(3) Nichtterminierung:
Es gibt eine unendliche Folge $\mathcal{R}_0, \mathcal{R}_1, \ldots$ von terminierenden, zu $\mathcal{E}$ äquivalenten Termersetzungssystem, die nicht konfluent sind.

Beispiel:

(1) Erfolg:
$\mathcal{E} := \{f(f(x,y), f(y,z)) \equiv y\}$ (zentrale Gruppoide)
$\mathcal{R}_0 = \{f(f(x,y), f(y,z)) \to y\}$

$CP(\mathcal{R}_0)$:

$$f\Big(f\big(f(x',y'), f(y',z')\big), f\big(f(y',z'), z\big)\Big) \to f\Big(y', f\big(f(y',z'), z\big)\Big)$$
$$\searrow \; \succ$$
$$f(y',z')$$

$$f\Big(f\big(y, f(x',y')\big), f\big(f(x',y'), f(y',z')\big)\Big) \to f\Big(f\big(x, f(x',y')\big), y'\Big)$$
$$\searrow \; \succ$$
$$f(x',y')$$

$$\mathcal{R}_1 = \mathcal{R}_0 \cup \\ \{f\Big(y', f\big(f(y', z'), z\big)\Big) \to f(y', z'), f\Big(f\big(x, f(x', y')\big), y'\Big) \to f(x', y')\}$$

(2) Fehlschlag:
$\mathcal{E} := \{f(x, g(y, z) \equiv g(f(x, y), f(x, z)), f(g(x, y), z) \equiv g(f(x, z), f(y, z))\}$
$\mathcal{R}_0 = \{f(x, g(y, z) \to g(f(x, y), f(x, z)), f(g(x, y), z) \to g(f(x, z), f(y, z))\}$
($\succ_{rpo}$: $f \sqsupset g$)

Bei der Normalformberechnung des ersten kritischen Paares von $\mathcal{R}_0$ ergibt sich, dass man die beiden Normalformen nicht orientieren kann $\rightsquigarrow$ "fail"

(3) Nichtterminierung:
$\mathcal{E} := \{f(g(f(x))) \equiv g(f(x))\}$
$\mathcal{R}_0 := \{f(g(f(x))) \to g(f(x))\}$
$\vdots$
$|\mathcal{R}_n| = |\{f(g^{n+1}(f(x))) \to g^{n+1}(f(x)) \mid n \in \mathbb{N}\}| = \infty$

Das letzte Beispiel zeigt, dass man unser Verfahren nur als *Semi-Entscheidungsverfahren* benutzen kann. Um $s \equiv_{\mathcal{E}} t$ zu überprüfen, testet man, ob s und t gemeinsame $\mathcal{R}_0$-Normalformen haben, wenn ja, dann gilt $s \equiv_{\mathcal{E}} t$. Wenn nein, dann berechne die $\mathcal{R}_1$-Normalformen usw. Wenn $s \equiv_{\mathcal{E}} t$ gilt, dann existiert auch ein $\mathcal{R}_n$, mit dem s und t gemeinsame Normalformen haben.

Satz 6.1 (Korrektheit von BASIC-COMPLETION**)** *Sei $\mathcal{E}$ ein endliches Gleichungssystem und $\succ$ eine Reduktionsordnung.*

(1) Falls BASIC-COMPLETION*$(\mathcal{E}, \succ)$ terminiert und $\mathcal{R}_n$ als Resultat liefert, dann ist $\mathcal{R}_n$ ein endliches, konvergentes und zu $\mathcal{E}$ äquivalentes Termersetzungssystem. Aus $\mathcal{R}_n$ lässt sich (mit* WORTPROBLEM*) ein Entscheidungsverfahren für das Wortproblem über $\mathcal{E}$ konstruieren.*

(2) Falls BASIC-COMPLETION*$(\mathcal{E}, \succ)$ nicht terminiert, dann ist $\mathcal{R}_\infty = \bigcup_{i \geq 0} \mathcal{R}_i$ ein unendliches, aber konvergentes und zu $\mathcal{E}$ äquivalentes Termersetzungssystem. In diesem Fall kann man den Vervollständigungsalgorithmus als Semi-Entscheidungsverfahren benutzen.*

Beweis:

(1)
- $\mathcal{R}_n$ ist endlich, weil $\mathcal{E}$ und damit auch $\mathcal{R}_0$ endlich ist, und da $CP(\mathcal{R}_i)$ endlich ist, wird $\mathcal{R}_i$ in jedem Schritt nur um endlich viele Regeln erweitert.
- Die Terminierung folgt, weil nur Regeln der Form $l \to r$ mit $l \succ r$ hinzugefügt werden. (Satz 4.5)
- Die Konfluenz folgt, da der Algorithmus nur dann $\mathcal{R}_n$ ausgibt, wenn alle kritischen Paare zusammenführbar sind, woraus die lokale Konfluenz (Satz 5.3) und damit die Konfluenz folgt (Satz 5.2).
- Äquivalenz von $\mathcal{R}_n$ und $\mathcal{E}$: Zeige $\forall i : \leftrightarrow^*_{\mathcal{E}} = \leftrightarrow^*_{\mathcal{R}_i}$

  I.A.: $i = 0$: $\leftrightarrow^*_{\mathcal{E}} = \leftrightarrow^*_{\mathcal{R}_0}$, da $\mathcal{R}_0$ nur $\mathcal{E}$ gerichtet ist
  I.S.: $\leftrightarrow^*_{\mathcal{E}} = \leftrightarrow^*_{\mathcal{R}_i} \subseteq \leftrightarrow^*_{\mathcal{R}_{i+1}}$
  $\forall \langle s, t \rangle \in CP(\mathcal{R}_i) : s' \leftarrow^* s \leftarrow p \to t \to^* t'$
  $\forall s' \to t'$ (bzw. $t' \to s'$) $\in \mathcal{R}_{i+1} : s' \leftrightarrow_{\mathcal{R}_i} t'$, d. h. $s' \leftrightarrow_{\mathcal{E}} t'$
  $\curvearrowright \leftrightarrow_{\mathcal{E}} = \leftrightarrow_{\mathcal{R}_{i+1}}$
  (Abgeschlossenheit unter Substitution und Kontexten)

(2)
- $\mathcal{R}_\infty$ ist unendlich, da in jedem Schritt neue Regeln hinzukommen
- Die Äquivalenz zu $\mathcal{E}$ und die Terminierung folgen wie in (1)
- Konfluenz von $\mathcal{R}_\infty$: Satz 5.3 und 5.2 gelten auch für unendliche Termersetzungssysteme. Zeige also: alle kritischen Paare von $\mathcal{R}_\infty$ sind zusammenführbar.
  Sei $\langle s,t\rangle \in CP(\mathcal{R}_\infty)$. Es existieren $l_1 \to r_1, l_2 \to r_2 \in \mathcal{R}_\infty$, deren Überlagerung ein kritisches Paar bildet. Da $\mathcal{R}_0 \subseteq \mathcal{R}_1 \subseteq \ldots$, existiert ein $\mathcal{R}_i$ mit $l_1 \to r_1, l_2 \to r_2 \in \mathcal{R}_i \curvearrowright \langle s,t\rangle \in CP(\mathcal{R}_i)$. In $\mathcal{R}_{i+1}$ ist $\langle s,t\rangle$ zusammenführbar und daher auch in $\mathcal{R}_\infty \supseteq \mathcal{R}_{i+1}$

q.e.d.

6.2 Ein verbesserter Vervollständigungsalgorithmus

Das Problem bei unserem bisherigen Vervollständigungsalgorithmus BASIC-COMPLETION ist, dass Regeln, die einmal hinzugenommen wurden, nicht mehr verändert oder gelöscht werden können. Es kann beispielweise vorkommen, dass mit neu entstandenen Regeln eine alte Regel überflüssig wird, wenn diese mit den neuen simuliert werden kann. Solche Regeln werden also nicht mehr gebraucht und sollten gelöscht werden können. D. h. neue Regeln sollten zur Vereinfachung der alten benutzt werden.

Unser Ziel in diesem Abschnitt wird es daher sein, *Transformationsregeln zur Vervollständigung* zu entwickeln. Je nach Anwendungsstrategie für die Regeln können dabei *unterschiedliche Vervollständigungsalgorithmen* entstehen. Die Transformationsregeln operieren auf Paaren von Gleichungs- und Termersetzungssystemen $(\mathcal{E},\mathcal{R})$. Wenn $(\mathcal{E},\mathcal{R})$ in $(\mathcal{E}',\mathcal{R}')$ überführt wird, dann soll gelten:

(1) $\leftrightarrow^*_{\mathcal{E}\cup\mathcal{R}} = \leftrightarrow^*_{\mathcal{E}'\cup\mathcal{R}'}$

(2) $\mathcal{R}$ terminiert $\curvearrowright$ $\mathcal{R}'$ terminiert

Das Vorgehen wird sein, dass man mit $(\mathcal{E},\emptyset)$ startet und solange transformiert bis man $(\emptyset,\mathcal{R})$ erhält.

Wegen (1) gilt: $\leftrightarrow^*_{\mathcal{E}} = \leftrightarrow^*_{\mathcal{R}}$ (d. h. $\mathcal{R}$ äquivalent zu $\mathcal{E}$)
Wegen (2) gilt: $\mathcal{R}$ terminiert.

Definition 6.1 (Transformationsregeln zur Vervollständigung) *Sei $\mathcal{E}$ ein Gleichungssystem, $\mathcal{R}$ ein Termersetzungssystem, $\succ$ eine Reduktionsordung. Dann definieren wir folgende* ***Transformationsregeln*** *auf Paaren $(\mathcal{E},\mathcal{R})$:*

(1) ***Generieren****:*

$$\frac{\mathcal{E},\mathcal{R}}{\mathcal{E}\cup\{s\equiv t\},\mathcal{R}} \qquad \textit{falls } \langle s,t\rangle \in CP(\mathcal{R})$$

(2) ***Orientieren****:*

$$\frac{\mathcal{E}\cup\{s\equiv t\},\mathcal{R}}{\mathcal{E},\mathcal{R}\cup\{s\to t\}} \qquad \textit{falls } s\succ t$$

$$\frac{\mathcal{E}\cup\{s\equiv t\},\mathcal{R}}{\mathcal{E},\mathcal{R}\cup\{t\to s\}} \qquad \textit{falls } t\succ s$$

(3) ***Löschen****:*

$$\frac{\mathcal{E}\cup\{s\equiv s\},\mathcal{R}}{\mathcal{E},\mathcal{R}}$$

(4) ***Gleichungs-Reduktion****:*

$$\frac{\mathcal{E} \cup \{s \equiv t\}, \mathcal{R}}{\mathcal{E} \cup \{u \equiv t\}, \mathcal{R}} \qquad \textit{falls } s \to_{\mathcal{R}} u$$

$$\frac{\mathcal{E} \cup \{s \equiv t\}, \mathcal{R}}{\mathcal{E} \cup \{s \equiv v\}, \mathcal{R}} \qquad \textit{falls } t \to_{\mathcal{R}} v$$

(5) ***Rechts-Reduktion****:*

$$\frac{\mathcal{E}, \mathcal{R} \cup \{s \to t\}}{\mathcal{E}, \mathcal{R} \cup \{s \to v\}} \qquad \textit{falls } t \to_{\mathcal{R}} v$$

(6) ***Links-Reduktion****:*

$$\frac{\mathcal{E}, \mathcal{R} \cup \{s \to t\}}{\mathcal{E} \cup \{u \equiv t\}, \mathcal{R}} \qquad \textit{falls } s \to_{\mathcal{R}} u \textit{ mit } l \to r$$

und l kann nicht mit $s \to t$ reduziert werden.

Man beachte, dass die beiden letzten Regeln (5) und (6) neu in Bezug auf BASIC-COMPLETION sind.

Schreibweise:

Wir schreiben $(\mathcal{E}, \mathcal{R}) \vdash_C (\mathcal{E}', \mathcal{R}')$, falls $\frac{\mathcal{E},\mathcal{R}}{\mathcal{E}',\mathcal{R}'}$

Beispiel:

$\mathcal{E} := \{h(x,y) \equiv f(x), h(x,y) \equiv f(y), g(x,y) \equiv h(x,y), g(x,y) \equiv a\}$

$$\begin{aligned}
& (\mathcal{E}, \emptyset) \\
\overset{(2)}{\vdash_C^+} \quad & (\emptyset, \{h(x,y) \to f(x), h(x,y) \to f(y), g(x,y) \to h(x,y), \\
& \qquad g(x,y) \to a\}) \\
\overset{(1)}{\vdash_C^+} \quad & (\{f(x) \equiv f(y), h(x,y) \equiv a\}, \{h(x,y) \to f(x), h(x,y) \to f(y), \\
& \qquad\qquad g(x,y) \to h(x,y), g(x,y) \to a\}) \\
\overset{(4)}{\vdash_C} \quad & (\{f(x) \equiv f(y), f(x) \equiv a\}, \{h(x,y) \to f(x), h(x,y) \to f(y), \\
& \qquad\qquad g(x,y) \to h(x,y), g(x,y) \to a\}) \\
\overset{(2)}{\vdash_C} \quad & (\{f(x) \equiv f(y)\}, \{h(x,y) \to f(x), h(x,y) \to f(y), g(x,y) \to h(x,y), \qquad (*) \\
& \qquad\qquad g(x,y) \to a, f(x) \to a\}) \\
\vdash_C^+ \quad & (\{a \equiv a\}, \{h(x,y) \to f(x), h(x,y) \to f(y), g(x,y) \to h(x,y), g(x,y) \to a, \\
& \qquad f(x) \to a\}) \\
\overset{(3)}{\vdash_C} \quad & (\emptyset, \{h(x,y) \to \underbrace{f(x)}_{a}, h(x,y) \to \underbrace{f(y)}_{a}, g(x,y) \to \underbrace{h(x,y)}_{a}, \\
& \qquad g(x,y) \to a, f(x) \to a\}) \\
\overset{(5)}{\vdash_C^+} \quad & (\emptyset, \{h(x,y) \to a, g(x,y) \to a, f(x) \to a\})
\end{aligned}$$

An der Stelle $(*)$ wäre BASIC-COMPLETION gescheitert.

Lemma 6.1 *Sei $\succ$ die bei der Transformation verwendete Reduktionsordung. Falls $l \succ r$ für alle $l \to r \in \mathcal{R}$ gilt und $(\mathcal{E}, \mathcal{R}) \vdash_C (\mathcal{E}', \mathcal{R}')$, dann gilt auch $l' \succ r'$ für alle $l' \to r' \in \mathcal{R}'$.*

Beweis:

Bei den Transformationsregeln (1), (3) und (4) gilt $\mathcal{R}' = \mathcal{R}$ ✓
Bei der Transformationsregel (6) gilt $\mathcal{R}' \subseteq \mathcal{R}$ ✓
Bei der Transformationsregel (2) ist es nach Definition klar ✓
Bei der Transformationsregel (5) gilt: $s \succ t$, $t \succ v$ (weil $\succ$ abgeschlossen unter Substitution und Kontexten) $\curvearrowright s \succ r$ (Transitivität) ✓

q.e.d.

Lemma 6.2 $(\mathcal{E}, \mathcal{R}) \vdash_C (\mathcal{E}', \mathcal{R}') \curvearrowright \leftrightarrow^*_{\mathcal{E} \cup \mathcal{R}} = \leftrightarrow^*_{\mathcal{E}' \cup \mathcal{R}'}$

Beweis:

Der Beweis ist klar für (2) und (3). Für die anderen Transformationregeln folgt das Lemma wegen der Abgeschlossenheit unter Substitution und Kontexten.

q.e.d.

Folgerung:

Falls $(\mathcal{E}, \emptyset) \vdash^*_C (\emptyset, \mathcal{R})$ und alle $\langle s, t \rangle \in CP(\mathcal{R})$ zusammenführbar sind, dann ist $\mathcal{R}$ ein konvergentes und zu $\mathcal{E}$ äquivalentes Termersetzungssystem.

Definition 6.2 *Wir definieren:*

- *Für eine endliche Folge $(\mathcal{E}_0, \mathcal{R}_0) \vdash^*_C (\mathcal{E}_n, \mathcal{R}_n)$ definieren wir die **persistenten Gleichungen** $\mathcal{E}_\omega := \mathcal{E}_n$ und die **persistenten Regeln** $\mathcal{R}_\omega := \mathcal{R}_n$.*
- *Für eine unendliche Folge $(\mathcal{E}_0, \mathcal{R}_0) \vdash_C (\mathcal{E}_1, \mathcal{R}_1) \vdash_C \ldots$ definieren wir die **persistenten Gleichungen** $\mathcal{E}_\omega := \bigcup_{i \geq 0} \bigcap_{j \geq i} \mathcal{E}_j$ und die **persistenten Regeln** $\mathcal{R}_\omega := \bigcup_{i \geq 0} \bigcap_{j \geq i} \mathcal{R}_j$.*

Definition 6.3 (Transformationsfolge, Vervollständigungsprozedur) *Eine endliche oder unendliche Transformationsfolge $(\mathcal{E}_0, \mathcal{R}_0) \vdash_C (\mathcal{E}_1, \mathcal{R}_1) \vdash_C \ldots$*

- *ist **erfolgreich** $:\curvearrowright \mathcal{E}_\omega = \emptyset$ und $\mathcal{R}_\omega$ ist ein konvergentes und zu $\mathcal{E}_0$ äquivalentes Termersetzungssystem.*
- ***schlägt fehl** $:\curvearrowright \mathcal{E}_\omega \neq \emptyset$*

*Eine **Vervollständigungsprozedur** ist ein Algorithmus, der als Eingabe ein Gleichungssystem $\mathcal{E}_0$ und eine Reduktionsordnung $\succ$ bekommt und damit eine Transformationsfolge $(\mathcal{E}_0, \mathcal{R}_0) \vdash_C (\mathcal{E}_1, \mathcal{R}_1) \vdash_C \ldots$ berechnet. Die Vervollständigungsprozedur ist **korrekt**, wenn für alle $\mathcal{E}_0$ und $\succ$ die berechnete Transformationsfolge erfolgreich ist oder fehlschlägt.*

Ohne Einschränkungen sind Vervollständigungsprozeduren nicht korrekt. Man muss verlangen, dass die Transformationsregeln so angewendet werden, dass "alle" kritsichen Paare berechnet werden.

Definition 6.4 *Eine endliche oder unendliche Transformationsfolge ist **fair** $:\curvearrowright CP(\mathcal{R}_\omega) \subseteq \bigcup_{i \geq 0} \mathcal{E}_i$, d. h. alle kritischen Paare der persistenten Regeln traten irgendwann mal auf.*

Satz 6.2 (Vervollständigungssatz) *Jede Vervollständigungsprozedur, bei der jede nicht-fehlschlagende Transformationsfolge fair ist, ist korrekt.*

Das letzte Ergebnis dieser Vorlesung erlaubt den Einsatz der Transformationsregeln als *Semi-Entscheidungsverfahren.*

Index

adäquat, 26
äquivalent, 25
Äquivalenzrelation, 15
Algebra, 8
allgemeingültig, 9

BASIC-COMPLETION, 60
Beweisrelation, 16
Birkhoff
    Satz von, 17

charakteristische Funktion, 45
CHURCH-ROSSER-Eigenschaft, 27, 29
CONFLUENCE, 56

Diamond-Lemma, 54
Domain, 11

Einbettungsordnung $\succ_{emb}$, 39
Erfüllbarkeit, 9
erfüllt, 9
Ersetzungsrelation, 16, 25
Ersetzungsschritt, 25

folgt, 9
fundiert, 27
funktionales Programm, 5
Funktionssymbol, 7

Generieren, 62
Gleichung, 8
    -ssystem, 8
Gleichungs-Reduktion, 63
Grundidentität, 19
Grundterm, 7

Halteproblem, 36
    universelles, 36
herleitbar, 16
Herleitung, 16

Interpretation, 8

König
    Lemma von, 36
kanonisch, *siehe* konvergent
konfluent, 29
    lokal, 53
Konfluenz, 29
    lokale, 53
KONGRUENZABSCHLUSS, 24
Kongruenzabschluss, 20
    bezüglich einer Menge von Termen, 22
Kongruenzrelation, 16
Konstante, 7
konvergent, 31
Konvergenz, 31
korrekt, 26, 64
kritisches Paar, 56
    Lemma, 56
    Menge aller, 56
Kruskal
    Satz von, 41

Löschen, 51, 62
Lösung, *siehe* Unifikator
Lemma von KÖNIG, 36
lexikographische Kombination, 41
lexikographische Pfadordnung, 42
    mit Status, 44
Links-Reduktion, 63

Manna
    Satz von, 40
matchen, 11
MATCHER, 53
Matcher, 11
Matching, 11
MGU, 50
Modell, 9
monoton, 14
Monotonie, 14
Multimenge, 45
Multimengen
    Menge aller, 45
Multimengen-Relation, 45

Ness
    Satz von, 40
Newman

Satz von, 54
NOETHERsches Induktionsprinzip, 35
Normalform, 27
eines Obejekts, 27
normalsierend, 27
eindeutig, 27

occur check, 50
Ordnung, 42
Orientieren, 62

persistente Gleichungen, 64
persistente Regeln, 64
Präzedenz, *siehe* Ordnung

Quotientenmenge, 18

Rechts-Reduktion, 63
Redex, 25
Reduktionsordnung, 40
Reduktionsrelation, 40
reduziert, 25
reflexiv, 15
reflexive Hülle, 15
Regel, 6, 25
rekursive Pfadordnung, 46
mit Status, 47
RIGHT-GROUND-TERMINIERUNG, 38

Satz von Birkhoff, 17
Satz von Kruskal, 41
Satz von Manna und Ness, 40
Satz von Newman, 54
Σ-Algebra, 8
Σ-Interpretation, 8
Signatur, 7
Simplifikationsordnung, 41
stabil, 12
Stabilität, 12
Stelle, 13
strukturelle Induktion
über Stellen, 14
über Terme, 12
Substitution, 11
allgemeinere, 49
Grund-, 11
identische, 11
Menge aller, 11
speziellere, 49
symmetrisch, 15
symmetrische Hülle, 16

Teilterm, 8, 21
direkter, 8
echter, 8
Teiltermmenge, 21
Term, 7
Termersetzungssystem (TES), 25
Termgleichung, *siehe* Gleichung
Terminierung, 28
Termreduktion, 51
Träger, 8
Transformationsfolge
erfolgreiche, 64
faire, 64
fehlschlagende, 64
Transformationsregeln
zur Vervollständigung, 62
transitiv, 15
transitiv-reflexiv-symmetrische Hülle, 16
transitiv-reflexive Hülle, 16
transitive Hülle, 15

unabhängig, 14
Unifikation, 49
Unifikationsproblem, 49
in gelöster Form, 50
Unifikationssatz, 53
Unifikator, 49
allgemeinerer, 49
Menge aller, 49
speziellerer, 49
unifizierbar, 49
UNIFY, 51
unterhalb, 14

Variable, 7
Variablenbelegung, 8
Variablenreduktion, 51
Variablenumbennung, 50
Vertauschen, 51
Vervollständigungsprozedur, 64
korrekte, 64

WORTPROBLEM, 31
Wortproblem, 10
Entscheidungsprozedur, 32

zusammenführbar, 29
Zusammenführbarkeit, 29