Termersetzungssysteme SS 2004

Prof. Dr. Jürgen Giesl
Mitgeschrieben von Timo Boettcher

Dieses Skript ist meine Mitschrift der Vorlesung Termersetzungssysteme im SS 2004 an der RWTH Aachen (Dozent: Prof. Dr. J. Giesl). Es handelt sich nicht um eine offizielle Veröffentlichung des "Lehr- und Forschungsgebiet Informatik II".
Ich übernehme keine Gewähr für die Fehlerfreiheit und Vollständigkeit des Skripts. Korrekturen und Ergänzungen bitte ich an timo_boettcher@post.rwth-aachen.de zu mailen.
Timo Boettcher, 30. Juli 2004

Inhaltsverzeichnis

1 Einführung **5**

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

3 Termersetzung und Deduktion von Gleichungen **11**
3.1 Deduktion von Gleichungen 11
3.2 Der Kongruenzabschluss bei Grundidentitäten 22
3.3 Termersetzungssysteme 28

4 Terminierung von Termersetzungssystemen **43**
4.1 Noethersche Induktion 43
4.2 Entscheidbarkeitsresultate zur Terminierung 45
4.3 Terminierung mit Reduktionsrelationen 48
4.4 Simplifikationsordnungen und rekursive Pfadordnungen 51

5 Konfluenz von Termersetzungssystemen **63**
5.1 Unifikation 63
5.2 Lokale Konfluenz und Kritische Paare 69

6 Vervollständigung von Termersetzungssystemen **81**
6.1 Der grundlegende Vervollständigungsalgorithmus 81
6.2 Ein vernesserter Vervollständigungsalgorithmus 85
6.3 Implizite Induktion 88

Kapitel 1

Einführung

[20.04.04]

Grundformalismus zum Rechnen und Beweisen mit Gleichungen

- Spezifikation von Programmen
- Programmanalyse und Programmverifikation
- Ausführung von Programmen
- Symbolisches Rechnen

Beispiel: Datentyp zur Darstellung natürlicher Zahlen:
Zwei Konstruktoren:
$\mathcal{O} \leftarrow$ repräsentiert 0
succ $\leftarrow$ "successor", repräsentiert Nachfolgerfunktion
$0 \widehat{=} \mathcal{O}$
$1 \widehat{=} succ(\mathcal{O})$
$2 \widehat{=} succ(succ(\mathcal{O}))$

Additionsalgorithmus:
$plus(\mathcal{O}, Y) \equiv Y$
$plus(succ(x), y) \equiv succ(plus(x, y))$

Berechnungen: "Anwendungen" der Gleichungen von links nach rechts
Berechne 2+1:

$$\begin{aligned} & plus(succ(succ(\mathcal{O})), succ(\mathcal{O})) \\ \rightarrow\ & succ(plus(succ(\mathcal{O}), succ(\mathcal{O}))) \\ \rightarrow\ & succ(succ(plus(\mathcal{O}, succ(\mathcal{O})))) \\ \rightarrow\ & succ(succ(succ(\mathcal{O}))) \widehat{=} 3 \end{aligned}$$

Systeme von Gleichungen wie oben bezeichnet man als Termersetzungssysteme

Schreibweise: "$\rightarrow$" statt "$\equiv$", um anzudeuten, das die Gleichungen von links nach rechts ausgewertet werden

Fragestellungen der Programmanalyse und Programmverifikation die man mit Termersetzungstechniken automatisch lösen will:

- Ist das Resultat eines Programmes immer eindeutig? (Konfluenz)
  $plus(\mathcal{O}, plus(succ(\mathcal{O}), \mathcal{O}))$

  $\rightarrow plus(succ(\mathcal{O}), \mathcal{O})$ oder $\rightarrow plus(\mathcal{O}, succ(plus(\mathcal{O}, \mathcal{O})))$

  $\rightarrow succ(\mathcal{O})$

  plus ist deterministisch, andere Programme nicht.

- Hält das Programm immer nach endlich vielen Schritten an? (Terminierung)

  plus terminiert, denn in jedem rekursiven Aufruf wird das 1. Argument kleiner

- Erfüllt das Programm seine Spezifikation? (Korrektheit)

  Gilt zum Beispiel $plus(succ(succ(\mathcal{O})), X) \equiv plus(succ(\mathcal{O}), succ(X))$, das heisst, folgt eine bestimmte Aussage aus bestimmten Axiomen?

Anderes Beispiel: Axiome zur Spezifikation von Gruppen:

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

Gilt i(i(n)) $\equiv$ n?
Kann man durch „Probieren" lösen. (lang und umständlich)
Mit Termersetzung kann man solche Beweise automatisch führen.
Wie kann man ein unvollständiges Programm automatisch vervollständigen?

Kapitel 2

Grundlagen

Sprache der Terme und der Gleichungen: (Teilmenge der) Prädikatenlogik 1. Stufe (Syntax und Semantik)

2.1 Syntax von Gleichungssytemen

Definition 2.1.1: Signatur (Alphabet, aus dem Ausdrücke unserer Sprache, also Terme, gebildet werden)
Eine Signatur $\Sigma = \cup_{i \in \mathbb{N}} \Sigma_i$ ist eine Vereinigung von paarweise disjunkten endlichen Mengen Σ_i. Jedes $f \in \Sigma_i$ heisst i-stelliges Funktionssymbol.
Elemente von Σ_0 heissen auch Konstanten. Wir verlangen stets, das Σ endlich ist und es mindestens eine Konstante gibt.

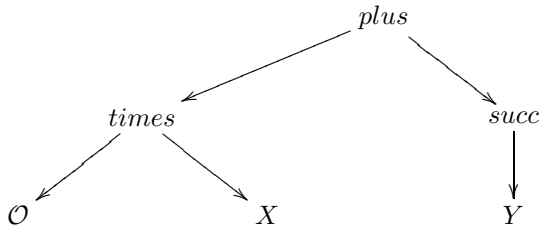
Beispiel 2.1.2:
$\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$,
$\Sigma_0 = \mathcal{O}$,
$\Sigma_1 = succ$,
$\Sigma_2 = plus, times$

Definition 2.1.3: Terme ("Datenobjekte")
Sei Σ eine Signatur, sei $\mathcal{V}$ eine nicht-leere unendliche Menge von Variablen mit $\Sigma \cap \mathcal{V} = \emptyset$. $\mathcal{T}(\Sigma, \mathcal{V})$ ist die Menge der Terme über Σ und $\mathcal{V}$, wobei $\mathcal{T}(\Sigma, \mathcal{V})$ die kleinste Teilmenge von $(\Sigma \cup \mathcal{V} \cup \{(,), \text{“}, \text{“}\}^*$ mit:

1. $V \subseteq \mathcal{T}(\Sigma, \mathcal{V})$
2. $f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, \mathcal{V})$ falls, $f \in \Sigma_n, t_i \in \mathcal{T}(\Sigma, \mathcal{V})$ für alle $i \in \{1, \ldots, n\}$

Falls $f \in \Sigma_0$, dann ist $f \in \mathcal{T}(\Sigma, \mathcal{V})$ Für einen Term t ist $\mathcal{V}(t)$ die Menge der Variablen in t und für $T \subseteq \mathcal{T}(\Sigma, \mathcal{V})$ ist $\mathcal{V}(T) = \cup_{t \in T} \mathcal{V}(t)$
$\mathcal{T}(\Sigma)$ steht für $\mathcal{T}(\Sigma, \emptyset)$ (Menge der Grundterme).

Beispiel 2.1.4:
Sei Σ wie in Beispiel 2.1.2 ($\Sigma = \{\mathcal{O}, succ, plus, times\}$)
$\mathcal{O}, succ(\mathcal{O}), plus(\mathcal{O}, succ(\mathcal{O})) \in \mathcal{T}(\Sigma)$
$plus(times(\mathcal{O}, \underbrace{X}_{\in\mathcal{V}}, succ(\underbrace{Y}_{\in\mathcal{V}}) \in \mathcal{T}(\Sigma, \mathcal{V})$
Terme lassen sich als Vielwegbäume darstellen:



Definition 2.1.3: (Fortsetzung)
Ein Term q ist Teilterm von t ($t \trianglerighteq q$) gdw. $t = q$, oder $t = f(t_1, \ldots, t_n)$ und $t_i \trianglerighteq q$ für ein $i \in \{1, \ldots, n\}$
Die Terme $t_1, \ldots, t_n$ sind direkte Teilterme von $f(t_1, \ldots, t_n)$. Ein Teilterm q von t ist echter Teilterm von $t (t \triangleright q)$ wenn $t \trianglerighteq q$ und $t \neq q$.

Beispiel 2.1.4: (Fortsetzung)
$t = plus(times(\mathcal{O}, X), succ(y))$
direkte Teilterme: $times(\mathcal{O}, X), succ(y)$. Andere echte Teilterme $\mathcal{O}, X, Y$

Definition 2.1.5: Gleichungen
Für $t_1, t_2 \in \mathcal{T}(\Sigma, \mathcal{V})$ heisst $t_1 \equiv t_2$ Gleichung (über Σ und $\mathcal{V}$). ("$\equiv$" bezeichnet Gleichheitssymbol)
Eine Menge $\mathcal{E}$ von Gleichungen heisst Gleichungssystem.
Mit Gleichungen kann man $\underbrace{\text{Programme}}_{\text{z.B. plus}}$, $\underbrace{\text{Datentypen}}_{\text{z.B. Gruppen}}$ etc. axiomatisieren.

2.2 Semantik von Gleichungssystemen

Zur Definition der Semantik versucht man Interpretationen, die eine Menge von Objekten $\mathcal{A}$ festlegen und jedem Funktionssymbol f eine Funktion $\mathcal{A}_f$ zuordnen, jeder Variable X ein Objekt $\beta(X) \in \mathcal{A}$ zuordnen.

Definition 2.2.1: Interpretation, Algebra
Für eine Signatur Σ ist eine Σ-Interpretation ein Tupel $I = (\mathcal{A}, \alpha, \beta)$. $\mathcal{A}$ ist der Träger der Interpretation mit $\mathcal{A} \neq \emptyset$, $\mathcal{A} = (\alpha_f)_f \in \Sigma$, wobei $\alpha_f : \mathcal{A} \times \mathcal{A} \times \cdots \times \mathcal{A} \to \mathcal{A}$ für $f \in \Sigma_n$

[23.04.04]

Beispiel 2.2.2:
$I = (\mathcal{A}, \alpha, \beta), \Sigma$ aus Beispiel 2.1.2, $\mathcal{V} = X, Y, \ldots$
$\mathcal{A} = \mathbb{N} = 0, 1, 2, \ldots$
$\alpha_0 = 0$

$$\left.\begin{aligned} \alpha_{succ}(n) &= n+1 \\ \alpha_{plus}(n,m) &= n+m \\ \alpha_{times}(n,m) &= n*m \end{aligned}\right\} (n, m \in \mathbb{N})$$

Definition 2.2.1: (Fortsetzung)
α_f heisst Deutung des Funktionssymbols f unter der Interpretation I.
Die Abbildung $\beta : \mathcal{V} \to \mathcal{A}$ heisst Variablenbelegung für die Interpretation I. Zu jeder Interpretation erhält man eine Funktion I: $\mathcal{T}(\Sigma, \mathcal{V}) \to \mathcal{A}$ wie folgt:
$I(x) = \beta(x)$ für alle $x \in \mathcal{V}$
$I(f(t_1, \ldots, t_n)) = \alpha_f(I(t_1), \ldots, I(t_n))$ für alle $f \in \Sigma_n$ und $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{V})$
I(t) heisst Interpretation des Terms t.
Eine Σ-Interpretation ohne Variablenbelegung $(\mathcal{A}, \alpha)$ heisst Σ-Algebra

Beispiel 2.2.2: (Fortsetzung)
$$I(plus(succ(x), y)) = \alpha_{plus}(\underbrace{\alpha_{succ}(\underbrace{\beta(x)}_{5})}_{6}, \underbrace{\beta(y)}_{3})$$
$I' = (\mathcal{A}, \alpha, \beta')$ mit $\beta'(x) = 2, \beta'(y) = 1$
$I'(plus(succ(x), y)) = (2+1)+1 = 4$

$I'' = (\mathbb{Q}, \alpha'', \beta)$ wobei
$$\alpha'' = 0$$
$$\alpha''_{succ}(n) = n+1$$
$$\alpha''_{plus}(n,m) = \begin{cases} \frac{n}{m}, & \text{falls } m \neq 0 \\ \text{sonst } 0 & (\alpha_f \text{ muss total sein}) \end{cases}$$

$\alpha''_{times}(n,m) = n*m$
$I''(plus(succ(x), y) = \frac{6}{3} = 2$

Definition 2.2.3: Erfüllen von Gleichungen, Modell
Eine Interpretation $I = (\mathcal{A}, \alpha, \beta)$ erfüllt eine Gleichung $t_1 \equiv t_2$ gdw. $I(t_1) = I(t_2)$
Eine Algebra $A = (\mathcal{A}, \alpha)$ erfüllt eine Gleichung $t_1 \equiv t_2$ gdw. $I(t_1) = I(t_2)$ für alle Interpretationen $I = (\mathcal{A}, \alpha, \beta)$
Für alle Variablenbelegungen β müssen die Deutungen von t_1 und t_2 also identisch sein. (Variablen sind implizit allquantifiziert).
Schreibweise: $I \vDash t_1 \equiv t_2$ bzw. $A \vDash t_1 \equiv t_2$.

Anstelle von Erfüllbarkeit spricht man auch von Modellen: $A \vDash t_1 \equiv t_2$ bedeutet „A ist Modell von $t_1 \equiv t_2$“
Algebra A ist Modell einer Gleichungsmenge $\mathcal{E}(A \vDash \mathcal{E})$ gdw. $A \vDash t_1 \equiv t_2$ für alle $t_1 \equiv t_2 \in \mathcal{E}$

Beispiel 2.2.4:
$I = (\mathbb{N}, \alpha, \beta) \quad (\beta(x) = 5, \beta(y) = 3)$
$I \vDash plus(x, y) \equiv succ(succ(succ(succ(succ(succ(succ(succ(\mathcal{O}))))))))$ (I Interpretiert beide Terme als 8)
$A = (\mathbb{N}, \alpha)$
$A \nvDash plus(x, y) \equiv succ(succ(succ(succ(succ(succ(succ(succ(\mathcal{O}))))))))$
$A \vDash plus(x, y) \equiv plus(y, x)$
$I'' \nvDash plus(x, y) \equiv plus(y, x)$
$(\alpha''_{plus}(n, m) = \frac{n}{m}$, falls $m \neq 0)$

Definition 2.2.3: (Fortsetzung)
Aus einer Gleichungsmenge $\mathcal{E}$ folgt die Gleichung $t_1 \equiv t_2$ ($\mathcal{E} \vDash t_1 \equiv t_2$) gdw. Für alle Algebren A mit $A \vDash \mathcal{E}$ auch $A \vDash t_1 \equiv t_2$ gilt. Statt $\emptyset \vDash t_1 \equiv t_2$ schreibt man auch $\vDash t_1 \equiv t_2$. Solche Gleichungen heissen allgemeingültig (alle Algebren sind Modelle von $t_1 \equiv t_2$).
“$\vDash$“ wird sowohl für Erfüllbarkeit als auch für Folgerbarkeit benutzt, was gemeint ist, erkennt man daran, ob links eine Interpretation/Algebra oder eine Gleichungsmenge steht.
Wir definieren die Relation $\equiv_{\mathcal{E}}$ auf $\mathcal{T}(\mathcal{E}, \mathcal{V}) \times \mathcal{T}(\Sigma, \mathcal{V})$ als $s \equiv_{\mathcal{E}} t$ gdw. $\mathcal{E} \vDash s \equiv t$.

Beispiel 2.2.4: (Fortsetzung)
$\{plus(x, y) \equiv plus(y, x)\} \vDash plus(\mathcal{O}, succ(\mathcal{O})) \equiv plus(succ(\mathcal{O}), \mathcal{O})$
Falls $\mathcal{E} = \{plus(x, y) \equiv plus(y, x)\}$ dann $plus(\mathcal{O}, succ(\mathcal{O})) \equiv_{\mathcal{E}} plus(succ(\mathcal{O}), \mathcal{O})$
Wäre $\mathcal{E} = Gruppenaxiome$, dann $i(i(n)) \equiv_{\mathcal{E}} n$

Die Frage, ob $s \equiv_{\mathcal{E}} t$ für zwei Terme s, t und ein Gleichungssystem $\mathcal{E}$ gilt, heisst Wortproblem.
Ziel: Löse Wortproblem (möglichst automatisch)

Kapitel 3

Termersetzung und Deduktion von Gleichungen

3.1 erstes naives Verfahren
3.2 Verfahren für den Spezialfall der Grundterme (Kongruenzabschluss)
3.3 Vefahren für den allgemeinen Fall (Termersetzungssysteme)

3.1 Deduktion von Gleichungen

Um $s \equiv_{\mathcal{E}} t$ zu untersuchen, müssen unendlich viele Algebren und unendlich viele Variablenbelegungen betrachtet werden.
$\equiv_{\mathcal{E}}$ ist semantisch definiert, ungeeignet für Automatisierung.
Ziel: Definiere eine syntaktische Relation, die $\equiv_{\mathcal{E}}$ entspricht, aber für die Automatisierung geeignet ist.
→ syntaktische Begriffe zur Manipulation von Termen.

Definition 3.1.1: Substitution, Matching
Eine Abbildung $\sigma : \mathcal{V} \to \mathcal{T}(\Sigma, \mathcal{V})$ mit $\sigma(x) \neq x$ für endlich viele Variablen x heisst Subsitution.
SUB$(\Sigma, \mathcal{V})$ ist die Menge aller Substitutionen. Die identische Substitution wird mit id bezeichnet (d.h. $id(x) = x$ für alle $x \in \mathcal{V}$)
$DOM(\sigma) = \{x \in \mathcal{V} | \sigma(x) \neq x\}$ heisst die Domain von σ. Da $DOM(\sigma)$ endlich ist, kann man jede Substitution als endliche Menge von Paaren darstellen: $\{x/\sigma(x) | x \in DOM(\sigma)\}$

Beispiel 3.1.2:
Σ wie in Beispiel 2.1.2
$\sigma(x) = plus(x, y)$, $\sigma(y) = \mathcal{O}$, $\sigma(z) = succ(z)$, $\sigma(u) = u$ für alle $u \in \mathcal{V} \backslash \{x, y, z\}$.
$DOM(\sigma) = \{x, y, z\}$
Schreibweise: $\sigma = \{x/plus(x, y),\ y/\mathcal{O},\ z/succ(z)\}$

Definition 3.1.1: (Fortsetzung)
Substitutionen werden homomorph zu Abbildungen auf Termen erweitert
$(\sigma : \mathcal{T}(\Sigma, \mathcal{V}) \to \mathcal{T}(\Sigma, \mathcal{V}))$, wobei:
$\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$.
Für zwei Terme s, t sagt man, dass s den Term t matcht gdw. es eine Substitution σ gibt mit $\sigma(s) = t$.
Die Substitution σ heisst Matcher von s und t.

Beispiel 3.1.2: (Fortsetzung)
$\sigma(plus(x, y)) = plus(\sigma(x), \sigma(y)) = plus(plus(x, y), \mathcal{O})$
Der Term plus(x,y) matcht den Term $plus(plus(x, y), \mathcal{O})$ und σ ist ein Matcher

Um $\equiv_\mathcal{E}$ auf syntaktische Weise nachzubilden, versuchen wir, die Eigenschaften von $\equiv_\mathcal{E}$ zu untersuchen.
Eine Eigenschaft $\equiv_\mathcal{E}$ ist stabil (d.h. abgeschlossen unter Subtitutionen)

Definition 3.1.3: Stabilität
Eine Relation $\to$ über Termen (d.h. $\to \subseteq \mathcal{T}(\Sigma, \mathcal{V}) \times \mathcal{T}(\Sigma, \mathcal{V})$) heisst stabil (oder abgeschlossen unter Substitution) gdw. für alle Terme t_1, t_2 und alle Substitutionen σ gilt:
Falls $t_1 \to t_2$, dann $\sigma(t_1) \to \sigma(t_2)$

[30.04.04]

Ziel: Ersetze die semantische Relation $\equiv_\mathcal{E}$ durch eine syntaktische Relation, die sich automatisch überprüfen lässt.
Hierzu: Untersuche die Eigenschaften von $\equiv_\mathcal{E}$.
Schreibweise für Substitution: Für $x^* = x_1, \ldots, x_n \in \mathcal{V}^*$ mit $x_i \neq x_j$ für $i \neq j$ und $t^* = \underbrace{t_1}_{\in \mathcal{V}}, \ldots, \underbrace{t_n}_{\in \mathcal{T}(\Sigma, \mathcal{V})^*}$ schreiben statt $\{x_1/t_1, \ldots, x_n/t_n\}$ auch $\{x^*/t^*\}$, statt "$\sigma(t)$" schreiben wir auch "$t\sigma$".
Stabilität: $plus(x, y) \to plus(\mathcal{O}, x)$
$\curvearrowright plus(s(\mathcal{O}), s(s(\mathcal{O}))) \to plus(\mathcal{O}, s(\mathcal{O}))$. Wir zeigen $\equiv_\mathcal{E}$ ist stabil.:
$t_1 \equiv_\mathcal{E} t_2 \curvearrowright t_1\sigma \equiv_\mathcal{E} t_2\sigma$ wenn: $plus(x, y) \equiv plus(\mathcal{O}, x) \in \mathcal{E}$
dann: $plus(s(\mathcal{O}), s(s(\mathcal{O}))) \equiv_\mathcal{E} plus(\mathcal{O}, s(\mathcal{O}))$
Beweisprinzip: strukturelle Induktion über Terme
zu Zeigen: $\varphi(t)$ für alle $t \in \mathcal{T}(\mathcal{E}, \mathcal{V})$
Induktionsanfang:

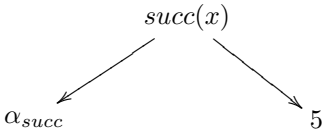
- t ist eine Variable $t \in \mathcal{V}$ zeige: $\varphi(t)$
- t ist eine Konstante $c \in \Sigma_0$ zeige: $\varphi(t)$

Induktionsschluss: $t = f(t_1, \ldots, t_n), n > 0$
zeige: $\underbrace{\varphi(t_1) \wedge \cdots \wedge \varphi(t_n)}_{\text{Induktionshypothese}} \to \underbrace{\varphi(f(t_1, \ldots, t_n))}_{\text{Induktionskonklusion}}$
Intuitiv: Beweisprinzip korrekt, da es direkt dem rekursiven Aufbau der Datenstruktur der Terme entspricht. Analoge Induktionsbeweise sind für alle rekursiv definierten Datenstrukturen möglich.

Formal: ist Spezialfall eines allgemeinen Beweisprinzips (noethersche Induktion)
$\curvearrowright$ Kapitel 4

Lemma 3.1.4: $\equiv_{\mathcal{E}}$ stabil
(Teil (a) zeigt Zusammenhang zwischen "Substitution" (syntaktisch) und "Variablenbelegung" (semantisch))
Sei $I = (\mathcal{A}, \alpha, \beta)$ eine Σ-Interpretation, seien $s, t \in \mathcal{T}(\Sigma, \mathcal{V})$, sei $\sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})$ und sei $I' = (\mathcal{A}, \alpha, \beta')$ mit $\beta'(x) = I(x\sigma)$ für alle $x \in \mathcal{V}$
z.B.

$$\begin{aligned} I &= (\mathbb{N}, \alpha, \beta), \\ \alpha_{succ}(n) &= n + 1, \\ \alpha_{plus}(n, m) &= n + m, \\ I' = (\mathbb{N}, \alpha, \beta') \text{ mit } \beta(x) &= 5 \\ \beta'(x) = I(\underbrace{x\sigma}_{succ(x)}) &= 6 \end{aligned}$$



a) $I(\underbrace{plus(x,x)\sigma}_{plus(s(x),s(x))}) = 12$
$I'(plus(\underbrace{x}_{6}, \underbrace{x}_{6})) = 12$

b) $I \vDash \underbrace{s^6(x)\sigma}_{\underbrace{s^7(x)}_{12}} = \underbrace{plus(x,x)\sigma}_{\underbrace{plus(s(x),s(x))}_{12}}$
$I' \vDash \underbrace{s^6(x)}_{12} \equiv \underbrace{plus(x,x)}_{12}$

c) $A \vDash plus(x, \mathcal{O}) \equiv x \curvearrowright A \vDash plus(s(x), \mathcal{O}) \equiv s(x)$

a) $I(t\sigma) = I'(t)$ (Substitutionslemma)

b) $I \vDash s\sigma \equiv t\sigma$ gdw. $I' \vDash s \equiv t$

c) Für alle Σ-Algebren A gilt: Wenn $A \vDash s \equiv t$, dann $A \vDash s\sigma \equiv t\sigma$

d) für alle Gleichungssysteme $\mathcal{E}$: Falls $s \equiv_{\mathcal{E}} t$, dann $s\sigma \equiv_{\mathcal{E}} t\sigma$
d.h.: $\equiv_{\mathcal{E}}$ ist stabil

Beweis:

a) Zeige $\varphi(t)$ durch strukturelle Induktion über t

$\varphi(t)$: „Für alle Substitutionen σ und alle Interpretationen I, I' wie oben gilt: $I(t\sigma) = I'(t)$“

Induktionsanfang: t ist eine Variable $x \in \mathcal{V}$:

$I(x\sigma)$

$\|$

$I'(x) = \beta'(x) = I(x\sigma)$

Vereinfachung: Betrachte den Fall $t = c \in \Sigma_0$ nicht extra, sondern nur den Fall $t = f(t_1, ..t_n)$, $n \geq 0$

zu Zeigen: $\underbrace{\varphi(t_1) \wedge \ldots \wedge \varphi(t_n)}_{\text{im Fall n=0 steht hier “nichts“, d.h. dort muss man } \varphi(f) \text{ direkt zeigen}} \rightarrow \varphi(f(t_1, \ldots, t_n))$

Induktionsschluss: $t = f(t_1, \ldots, t_n), n \geq 0$

Induktionshypothese: $\varphi(t_1)$ für alle $1 \leq i \leq n$ „Für alle $\sigma, I, I' : I(t_i\sigma) = I'(t_i)$´´

$I(t\sigma) = I(f(t_1, \ldots, t_n)\sigma) = I(f(t_1\sigma, \ldots, t_n\sigma)) = \alpha_f(I(t_1\sigma), \ldots, I(t_n\sigma)) \overset{I.H.}{=} \alpha_f(I'(t_1), \ldots, I'(t_n)) = I'(f(t_1, \ldots, t_n)) = I'(t)$

b) $I \vDash s\sigma \equiv t\sigma$ gdw. $I(s\sigma) = I(t\sigma)$ gdw. $I(s\sigma) = I(s\sigma)$ gdw. $I'(s) = I'(t)$ gdw. $I' \vDash s \equiv t$

c) Sei $A = (\mathcal{A}, \alpha)$ mit $A \vDash s \equiv t$.

Zu Zeigen ist: $A \vDash s\sigma \equiv t\sigma$, d.h. für alle Interpretationen $I = (\mathcal{A}, \alpha, \beta)$ gilt: $I \vDash s\sigma \equiv t\sigma$

Nach (b) genügt es zu zeigen, dass $I' \vDash s \equiv t$, wobei $I'(\mathcal{A}, \alpha, \beta')$ mit $\beta'(x) = I(x\sigma)$ für alle $x \in \mathcal{V}$

Aus $A \vDash s \equiv t$ folgt aber $I' \vDash s \equiv t$, da I' und A gleichen Träger $\mathcal{A}$ und gleiche Deutungsfunktion α haben.

d) Sei $s \equiv_{\mathcal{E}} t$, d.h. $\mathcal{E} \vDash s \equiv t$. Das bedeutet: Für alle Algebren A mit $A \vDash \mathcal{E}$ gilt: $A \vDash s \equiv t$.

$\overset{(c)}{\curvearrowright}$ Für alle Algebren A mit $A \vDash \mathcal{E}$ gilt $A \vDash s\sigma \equiv t\sigma \curvearrowright s\sigma \equiv_{\mathcal{E}} t\sigma$

□

Definition 3.1.5: Stelle

Für einen Term t ist Occ(t) die Menge aller Stellen von t, wobei Occ(t) die kleinste Teilmenge von $\mathbb{N}^*$ ist mit:

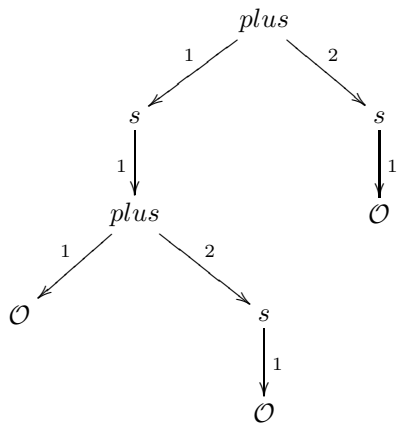
- $\epsilon \in Occ(t)$

- $i\pi \in Occ(t)$, falls $t = f(t_1, \ldots, t_n), 1 \leq i \leq n, \pi \in Occ(t_i)$

Für einen Term t und $\pi \in Occ(t)$ bezeichnet $t|_\pi$ den Teilterm von t an der Stelle π, wobei

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

Beispiel 3.1.6:
$t = plus(s(plus(\mathcal{O}, s(\mathcal{O}))), s(\mathcal{O}))$,



$t|_\epsilon = t$
$t|_1 = s(plus(\mathcal{O}, s(\mathcal{O})))$
$t|_{112} = s(\mathcal{O})$
$t|_{21} = \mathcal{O}$

Definition 3.1.5: (Fortsetzung)
Für t und $\pi \in Occ(t)$ und Term v bezeichnet $t[v]_\pi$ den Term, der aus t entsteht, indem $t|_\pi$ durch v ersetzt wird, d.h.

- $t[v]_\epsilon = v$
- $f(t_1, \ldots, t_n)[v]_{i\pi} = f(t_1, \ldots, t_{i-1}, t_i[v]_\pi, t_{i+1}, \ldots, t_n)$

Beispiel 3.1.6: (Fortsetzung)
$t[times(\mathcal{O}, \mathcal{O})]_{112} = plus(s(plus(\mathcal{O}, times(\mathcal{O}, \mathcal{O}))), s(\mathcal{O}))$

Definition 3.1.5: (Fortsetzung)
Für $\pi_1, \pi_2 \in Occ(t)$ ist π_1 unterhalb von π_2 ($\pi_1 >_{\mathbb{N}^*} \pi_2$) gdw. π_2 ein echtes Anfangsstück von π_1 ist (d.h.: $\pi_1 = \pi_2\pi$ für ein $\pi \in \mathbb{N}^+$). π_1 und π_2 sind voneinander unabhängig ($\pi_1 \perp \pi_2$) gdw. weder $\pi_1 >_{\mathbb{N}^*} \pi_2$ noch $\pi_2 >_{\mathbb{N}^*} \pi_1$

Beispiel: $112 >_{\mathbb{N}^*} 11 >_{\mathbb{N}^*} 1 >_{\mathbb{N}^*} \epsilon, 111 \perp 112$

Definition 3.1.7: Monotonie
Eine Relation $\rightarrow$ über Termen heisst monoton gdw. für alle Terme t_1, t_2, q und alle $\pi \in Occ(q)$ gilt:
Wenn $t_1 \rightarrow t_2$, dann $q[t_1]_\pi \rightarrow q[t_2]_\pi$
Beispiel: $plus(x, \mathcal{O}) \rightarrow x \curvearrowright s(plus(x, \mathcal{O})) \rightarrow s(x)$
Strukturelle Induktion ist auch über Stellen π möglich.

[07.05.04]

Lemma 3.1.8: $\equiv_\mathcal{E}$ monoton
Seien $s, t, q \in \mathcal{T}(\Sigma, \mathcal{V})$ und $\pi \in Occ(q)$. Dann gilt:

a) Für alle Σ-Algebren A gilt: Falls $A \vDash s \equiv t$, dann $A \vDash q[s]_\pi \equiv q[t]_\pi$

b) Für alle Gleichungssysteme $\mathcal{E}$ gilt: Falls $s \equiv_\mathcal{E} t$, dann $q[s]_\pi \equiv_\mathcal{E} q[t]_\pi$, d.h. $\equiv_\mathcal{E}$ ist monoton.

Beweis:

a) Strukturelle Induktion über π. Die Aussage $\varphi(\pi)$, die wir beweisen wollen, lautet:

$\varphi(\pi)$: „Für alle Algebren A und alle Terme s, t, q mit $\pi \in Occ(q)$ gilt: $A \vDash s \equiv t \curvearrowright A \vDash q[s]_\pi \equiv q[t]_\pi$."

Induktionsanfang: $\pi = \epsilon \quad q[s]_\epsilon = s, q[t]_\epsilon = t$

$\curvearrowright A \vDash s \equiv t \curvearrowright A \vDash \underbrace{q[s]_\epsilon}_{s} \equiv \underbrace{q[s]_\epsilon}_{t}$

Induktionsschluss: $\pi = i\pi'$

Da $i\pi' \in Occ(q)$, hat q die Gestalt $f(q_1, \ldots, q_i, \ldots, q_n)$ und $\pi' \in Occ(q_i)$.

$q[s]_{i\pi'} = f(q_1, \ldots, q_i[s]_{\pi'}, \ldots, q_n)$

$q[t]_{i\pi'} = f(q_1, \ldots, q_i[t]_{\pi'}, \ldots, q_n)$

Sei $A = (\mathcal{A}, \alpha)$ mit $A \vDash s \equiv t$. Zu zeigen ist, dass für alle Interpretationen $I = (\mathcal{A}, \alpha, \beta)$ mit beliebigen β gilt:

$I \vDash f(q_1, \ldots, q_i[s]_{\pi'}, \ldots, q_n) \equiv f(q_1, \ldots, q_i[t]_{\pi'}, \ldots, q_n)$

d.h.: $\alpha_f(I(q_1), \ldots, I(q_i[s]_{\pi'}), \ldots, I(q_n)) =$

$\alpha_f(I(q_1), \ldots, I(q_i[t]_{\pi'}), \ldots, I(q_n))$

Hierfür genügt es zu zeigen, dass $I(q_i[s]_{\pi'}) = I(q_i[t]_{\pi'})$.

Induktionshypothese: $\varphi(\pi')$: „Für alle A, s, t, $\bar{q}$ mit $\pi' \in Occ(\bar{q})$ gilt:

$A \vDash s \equiv t \curvearrowright A \vDash \bar{q}[s]_{\pi'} \equiv \bar{q}[t]_{\pi'}$."

Da die Induktionshypothese für alle $\bar{q}$ zur Verfügung steht, steht sie auch für $\bar{q} := q_i$ zur Verfügung.

D.h.: Da $\pi' \in Occ(q_i)$, gilt aufgrund der Induktionshypothese:

$A \vDash q_0[s]_{\pi'} \equiv q_i[t]_{\pi'} \curvearrowright I \vDash q_i[s]_{\pi'} \equiv q_i[t]_{\pi'} \curvearrowright (*)$ gilt.

b) $s \equiv_{\mathcal{E}} t \curvearrowright \mathcal{E} \vDash s \equiv t \curvearrowright$ für alle Algebren A:
$A \vDash \mathcal{E} \curvearrowright A \vDash s \equiv t$
$\overset{(a)}{\curvearrowright}$ für alle Algebren A: $A \vDash \mathcal{E} \curvearrowright A \vDash q[s]_\pi \equiv q[t]_\pi$
$\curvearrowright q[s]_\pi \equiv_{\mathcal{E}} q[t]_\pi$
$\equiv_{\mathcal{E}}$ stabil und monoton

□

Definition 3.1.9: Äquivalenzrelation, Kongruenzrelation
Sei M eine Menge und sei $\rightarrow$ eine Relation auf M ($\rightarrow \subseteq M \times M$). Man sagt auch, (M, $\rightarrow$) ist ein abstraktes Reduktionssystem (ARS).
Die Relation „$\rightarrow$“ heisst

- reflexiv, falls t $\rightarrow$ t für alle $t \in M$ gilt (Bsp:. $\geq$ auf $\mathbb{N}$, $\trianglerighteq$ (Teiltermrelation), $=$ auf $\mathbb{N}$, ...)
- symmetrisch, falls aus $t_1 \rightarrow t_2$ jeweils $t_2 \rightarrow t_1$ folgt, für alle $t_1, t_2 \in M$ (Bsp.: $=$ auf $\mathbb{N}$)
- transitiv, falls aus $t_1 \rightarrow t_2$ und $t_2 \rightarrow t_3$ jeweils $t_1 \rightarrow t_3$ folgt, für alle $t_1, t_2, t_3 \in M$ (Bsp.: $\geq$ auf $\mathbb{N}$, $\trianglerighteq$, $=$)
- Äquivalenzrelation, falls $\rightarrow$ reflexiv, symmetrisch und transitiv ist (Bsp.: $=$ auf $\mathbb{N}$)

Die reflexive Hülle von $\rightarrow$ (Bezeichnung: $\rightarrow^=$) ist die kleinste reflexive Relation, die $\rightarrow$ enthält, d.h.:

- wenn $t_1 \rightarrow t_2$, dann $t_1 \rightarrow^= t_2$
- $t \rightarrow^= t$

($>^=$=$\geq$, $\rhd^= = \trianglerighteq$)
Die transitive Hülle $\rightarrow^+$ ist die kleinste transitive Relation, die $\rightarrow$ enthält, d.h.:

- wenn $t_1 \rightarrow t_2$, dann $t_1 \rightarrow^+ t_2$
- wenn $t_1 \rightarrow t_2 \rightarrow^+ t_3$, dann $t_1 \rightarrow^+ t_3$
  ($s \rightarrow t$ bedeutet also, dass es $s_0, \ldots, s_n$ mit $n > 0$ gibt, mit $s = s_0 \rightarrow s_1 \rightarrow \ldots \rightarrow s_n = t$)
  (Bsp.: $\rightarrow \subseteq \mathbb{N} \times \mathbb{N}, x \rightarrow y$ gdw. $x = y + 1$ $5 \rightarrow 4$, $4 \rightarrow 3$

Die transitiv-reflexive Hülle $\rightarrow^*$ ist die kleinste transitive und reflexive Relation, die $\rightarrow$ enthält, d.h.:

- wenn $t_1 \rightarrow^+ t_2$, dann $t_1 \rightarrow^* t_2$
- $t \rightarrow^* t$

Die Relation $\leftrightarrow$ ist die symmetrische Hülle von $\rightarrow$, d.h. es gilt $t_1 \leftrightarrow t_2$ gdw. $t_1 \rightarrow t_2$ oder $t_2 \rightarrow t_1$. Die Relation $\leftrightarrow^*$ ist also die kleinste Äquivalenzrelation, die $\rightarrow$ enthält.
Eine Äquivalenzrelation $\rightarrow$ über Termen $\mathcal{T}(\Sigma, \mathcal{V})$ heisst Kongruenzrelation, wenn für alle $f \in \Sigma$ und alle $s_1, \ldots, s_n, t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{V})$ mit $s_1 \rightarrow t_1, \ldots, s_n \rightarrow t_n$ folgt: $f(s_1, \ldots, s_n) \rightarrow f(t_1, \ldots, t_n)$

Lemma 3.1.10: $\equiv_\mathcal{E}$ ist Kongruenzrelation
Für alle Gleichungssysteme $\mathcal{E}$ ist $\equiv_\mathcal{E}$ eine Äquivalenz- und Kongruenzrelation.
Beweis: Übung
Beweisidee:
Äquivalenz: ($A \vDash t \equiv t, A \vDash s \equiv t \curvearrowright A \vDash t \equiv s$,
$A \vDash s \equiv t$ und $A \vDash t \equiv r \curvearrowright A \vDash s \equiv r$)
Kongruenz: $\equiv_\mathcal{E}$ monoton; Jede monotone Äquivalenzrelation ist eine Kongruenzrelation.

Ziel: $s \equiv_\mathcal{E} t$ automatisch beweisen. Dann: Ersetze semantische Relation $\equiv_\mathcal{E}$ durch eine syntaktische Relation, die automatisch überprüft werden kann.

Definition 3.1.11: Ersetzungsrelation
Für ein Gleichungssystem $\mathcal{E}$ ist die Ersetzungsrelation $\rightarrow_\mathcal{E} \subseteq \mathcal{T}(\Sigma, \mathcal{V}) \times \mathcal{T}(\Sigma, \mathcal{V})$ definiert als:
$s \rightarrow_\mathcal{E}$ gdw. $s|_\pi = t_1\sigma$ und $t = s[t_2\sigma]_\pi$
für eine Stelle $\pi \in Occ(s)$, eine Gleichung $t_1 \equiv t_2 \in \mathcal{E}$ und eine Substitution $\sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})$
(Bsp.: $plus(x, \mathcal{O}) \equiv x \in \mathcal{E}$
$plus(\mathcal{O}, \mathcal{O}) \rightarrow_\mathcal{E} \mathcal{O}, s(plus(\mathcal{O}, \mathcal{O})) \rightarrow_\mathcal{E} s(\mathcal{O})$)
Die Relation $\leftrightarrow^*_\mathcal{E}$ heisst Beweisrelation von $\mathcal{E}$
($s \leftrightarrow^*_\mathcal{E} t$ falls $s = s_0 \leftrightarrow_\mathcal{E} s_1 \leftrightarrow_\mathcal{E} \ldots \leftrightarrow_\mathcal{E} s_n = t$).
Wir sagen $s \equiv t$ ist aus $\mathcal{E}$ herleitbar („$\mathcal{E} \vdash s \equiv t$“) gdw. $s \leftrightarrow^*_\mathcal{E} t$
D.h.: $\underbrace{\mathcal{E} \vDash s \equiv t}_{\text{Semantik}(s \equiv_\mathcal{E} t)}$ $\underbrace{\text{gdw.}}_{\text{(Satz von Birkhoff)}}$ $\underbrace{\mathcal{E} \vdash s \equiv t}_{\text{Syntax}(\leftrightarrow^*_\mathcal{E} t)}$

Beispiel 3.1.12:
$\mathcal{E} = \{plus(\mathcal{O}, y) \equiv y, plus(succ(x), y) \equiv succ(plus(x, y))\}$
Gilt: $plus(s(s(\mathcal{O})), x) \equiv_\mathcal{E} plus(s(\mathcal{O}), s(x))$?
Es gilt: $plus(s(s(\mathcal{O})), x) \Leftrightarrow^*_\mathcal{E} plus(s(\mathcal{O}), s(x))$
[Folie]
Erstes automatisierbares Verfahren, um das Wortproblem zu lösen.

Lemma 3.1.13: $\leftrightarrow^*_\mathcal{E}$ stabile und monotone Kongruenzrelation
Sei $\mathcal{E}$ ein Gleichungssystem, $s, t, q \in \mathcal{T}(\Sigma, \mathcal{V}), \pi \in Occ(q), \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})$ Dann ist $\leftrightarrow^*_\mathcal{E}$ eine Äquivalenz- und Kongruenzrelation und es gilt:

a)

$$\left.\begin{array}{l}\text{Falls } \to_{\mathcal{E}} t, \text{ dann } s\sigma \to_{\mathcal{E}} t\sigma \\ \text{Falls } \leftrightarrow^*_{\mathcal{E}} t, \text{ dann } s\sigma \leftrightarrow^*_{\mathcal{E}} t\sigma\end{array}\right\} \text{d.h. } \to_{\mathcal{E}} \text{ und } \leftrightarrow^+_{\mathcal{E}} \text{ sind stabil.}$$

b)

$$\left.\begin{array}{l}\text{Falls } \to_{\mathcal{E}} t, \text{ dann } sq[s]_\pi \to_{\mathcal{E}} q[t]_\pi \\ \text{Falls } \leftrightarrow^*_{\mathcal{E}} t, \text{ dann } sq[s]_\pi \leftrightarrow^*_{\mathcal{E}} q[t]_\pi\end{array}\right\} \text{d.h. } \to_{\mathcal{E}} \text{ und } \leftrightarrow^+_{\mathcal{E}} \text{ sind monoton.}$$

Beweis: Übung
Schritt von $\to_{\mathcal{E}}$ nach $\leftrightarrow^*_{\mathcal{E}}$: Induktion über Anzahl der $\leftrightarrow_{\mathcal{E}}$ Schritte

[11.05.04]

Wortproblem: $s \equiv_{\mathcal{E}} t (\mathcal{E} \vDash s \equiv t)$ Eigenschaften von $\equiv_{\mathcal{E}}$: stabil, monoton, Äquivalenzrelation, Kongruenzrelation
Beweisrelation: $s \leftrightarrow^*_{\mathcal{E}} t (\mathcal{E} \vdash s \equiv t)$ (Syntax)

Satz 3.1.14: Birkhoff, 1935
Sei $\mathcal{E}$ ein Gleichungssystem. Dann sind die Relationen $\equiv_{\mathcal{E}}$ und $\leftrightarrow^*_{\mathcal{E}}$ identisch, d.h. es gilt $\mathcal{E} \vDash s \equiv t$ gdw. $\mathcal{E} \vdash s \equiv t$ für alle Terme s und t.
Beweis: Korrektheit $(s \leftrightarrow^*_{\mathcal{E}} t \frown s \equiv_{\mathcal{E}} t)$
Zeige zunächst: $s \leftrightarrow_{\mathcal{E}} t \frown s \equiv_{\mathcal{E}} t$ (*)
Es existiert eine Gleichung $t_1 \equiv t_2$ oder $t_2 \equiv t_1$ in $\mathcal{E}$, so dass $s|_\pi = t_1\sigma$ und $t = s[t_2\sigma]_\pi$ für ein $\pi \in Occ(s)$ une eine Substitution σ.
Es gilt: $t_1 \equiv_{\mathcal{E}} t_2$ (wegen der Symmetrie von $\equiv_{\mathcal{E}}$, Lemma 3.1.10)
$t_1\sigma \equiv_{\mathcal{E}} t_2\sigma$ (wegen Stabilität von $\equiv_{\mathcal{E}}$, Lemma 3.1.4(d))
$\underbrace{s[t_1\sigma]_\pi}_{s} \equiv_{\mathcal{E}} \underbrace{s[t_2\sigma]_\pi}_{t}$ (wegen Monotonie von $\equiv_{\mathcal{E}}$, Lemma 3.1.8 (b))
somit gilt (*)
Zeigen nun $s \leftrightarrow^*_{\mathcal{E}} t \frown s \equiv_{\mathcal{E}} t$, d.h. für alle $n \in \mathbb{N}$: $s \leftrightarrow^n_{\mathcal{E}} t \frown s \equiv_{\mathcal{E}} t$
Induktion über n ($\widehat{=}$ Induktion über die Länge der Herleitung $s \leftrightarrow^*_{\mathcal{E}} t$)
Induktionsanfang: $n = 0$
$s \leftrightarrow^0_{\mathcal{E}} t \frown s = t \frown s \equiv_{\mathcal{E}} t$ (wegen der Reflexivität von $\equiv_{\mathcal{E}}$, Lemma 3.1.10)
Induktionsschluss: $n > 0$
$s = s_0 \leftrightarrow_{\mathcal{E}} s_1 \leftrightarrow_{\mathcal{E}} s_2 \leftrightarrow_{\mathcal{E}} \ldots \leftrightarrow_{\mathcal{E}} s_{n-1} \leftrightarrow_{\mathcal{E}} s_n = t$

$$\left.\begin{array}{r}\text{Induktionshypothese: } s \leftrightarrow^{n-1}_{\mathcal{E}} s_{n-1} \frown s \equiv_{\mathcal{E}} s_{n-1} \\ \text{wegen (*): } s_{n-1} \leftrightarrow_{\mathcal{E}} t \frown s_{n-1} \equiv_{\mathcal{E}} t\end{array}\right\} \begin{array}{l} s \equiv_{\mathcal{E}} t \\ \text{(wegen Transitivität} \\ \text{von } \equiv_{\mathcal{E}} \text{, Lemma 3.1.10)}\end{array}$$

Vollständigkeit $s \equiv_{\mathcal{E}} t \frown s \leftrightarrow^*_{\mathcal{E}} t$
$s \equiv_{\mathcal{E}} t$ bedeutet: $A \vDash \mathcal{E} \frown A \vDash s \equiv t$
Vorgehen:

1. Definiere eine Interpretation $I = (\mathcal{A}, \alpha, \beta)$, so dass $I \vDash s \equiv t$ gdw $s \leftrightarrow^*_{\mathcal{E}} t$
2. Zeige, dass $A = (\mathcal{A}, \alpha)$ ein Modell von $\mathcal{E}$ ist, d.h. $A \vDash \mathcal{E}$

$\curvearrowright$ Dann ist die Vollständigkeit bewiesen:
$s \equiv_{\mathcal{E}} t \curvearrowright A \vDash s \equiv t$ (denn $A \vDash \mathcal{E}$ nach (2)) $\curvearrowright I \vDash s \equiv t \curvearrowright s \leftrightarrow_{\mathcal{E}}^* t$ (nach (1))
Schritt (1):
1.Idee: $A = \mathcal{T}(\Sigma, \mathcal{V})$, Interpretiere jeden Term t als „sich selbst“ $\alpha_f = f$, $\beta(x) = x$
Dann gilt: $I(t) = t$ für alle Terme t (strukturelle Induktion über t)
(Falls $\mathcal{E}$ die plus-Gleichungen sind, dann: $plus(\mathcal{O}, \mathcal{O}) \leftrightarrow_{\mathcal{E}} \mathcal{O}$
$I \nvDash plus(\mathcal{O}, \mathcal{O}) \equiv \mathcal{O}$, denn $\underbrace{I(plus(\mathcal{O}, \mathcal{O}))}_{plus(\mathcal{O},\mathcal{O})} \neq \underbrace{I(\mathcal{O}))}_{\mathcal{O}}$
2.Idee: Wähle Träger A anders: Wir wissen, dass $\leftrightarrow_{\mathcal{E}}^*$ eine Äquivalenzrelation ist (Lemma 3.1.13).
$[s]_{\leftrightarrow_{\mathcal{E}}^*}$ ist die Äquivalenzklasse von s, wobei $[s]_{\leftrightarrow_{\mathcal{E}}^*} = t | s \leftrightarrow_{\mathcal{E}}^* t$
Da $\leftrightarrow_{\mathcal{E}}^*$ eine Äquivalenzrelation ist, sind zwei Äquivalenzklassen $[s]_{\leftrightarrow_{\mathcal{E}}^*}$ und $[t]_{\leftrightarrow_{\mathcal{E}}^*}$ entweder identisch oder disjunkt.
$\curvearrowright$ Die Menge der Terme $\mathcal{T}(\Sigma, \mathcal{V})$ zerfällt also in mehrer disjunkte Äquivalenzklassen. Die Menge dieser Äquivalenzklassen ist die Quotientenmenge $\mathcal{T}(\Sigma, \mathcal{V})/_{\leftrightarrow_{\mathcal{E}}^*} = [s]_{\{} \leftrightarrow_{\mathcal{E}}^* | s \in \mathcal{T}(\Sigma, \mathcal{V})\}$
(Beispiel: $\mathcal{E}_{plus} \mathcal{T}(\Sigma, \mathcal{V})/_{\leftrightarrow_{\mathcal{E}}^*} = \{[0]_{\leftrightarrow_{\mathcal{E}}^*}, [s(\mathcal{O})]_{\leftrightarrow_{\mathcal{E}}^*}, \ldots, [x]_{\leftrightarrow_{\mathcal{E}}^*}, [y]_{\leftrightarrow_{\mathcal{E}}^*}, \ldots, [plus(x, y)]_{\leftrightarrow_{\mathcal{E}}^*}, \ldots\}$
$[\mathcal{O}]_{\leftrightarrow_{\mathcal{E}}^*} = \{\mathcal{O}, plus(\mathcal{O}, \mathcal{O}), plus(plus(\mathcal{O}, \mathcal{O}), \mathcal{O}), \ldots\}$
$[X]_{\leftrightarrow_{\mathcal{E}}^*} = \{x, plus(\mathcal{O}, x), \ldots\})$
Definiere I so, dass $\mathcal{A} = \mathcal{T}(\Sigma, \mathcal{V})/_{\leftrightarrow_{\mathcal{E}}^*}$ und $I(t) = [t]_{\leftrightarrow_{\mathcal{E}}^*}$ (jeder Term soll als seine Äquivalenzklasse interpretiert werden)
Dann gilt $I \vDash s \equiv t$ gdw. $I(s) = I(t)$ gdw. $[s]_{\leftrightarrow_{\mathcal{E}}^*} = [t]_{\leftrightarrow_{\mathcal{E}}^*}$ gdw. $s_{\leftrightarrow_{\mathcal{E}}^* t}$
α_f: Wir wollen, dass $I(f(t_1, \ldots, t_n)) = \alpha_f(I(t_1), \ldots, I(t_n)) = \alpha_f([t_1], \ldots, [t_n]) \stackrel{!}{=} [f(t_1, \ldots, t_n)]$
Definiere daher: $\alpha_f([t_1]_{\leftrightarrow_{\mathcal{E}}^*}, .., [t_n]_{\leftrightarrow_{\mathcal{E}}^*}) = [f(\underbrace{t_1}, \ldots, t_n)]_{\leftrightarrow_{\mathcal{E}}^*}$

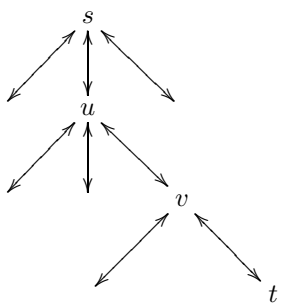
irgendein Term aus $[t_1]_{\leftrightarrow_{\mathcal{E}}^*}$, man könnte auch einen beliebigen Term $s_1 \in [t_1]$ nehmen, denn aus $s_1 \leftrightarrow_{\mathcal{E}}^* t_1 \curvearrowright f(s_1, \ldots) \leftrightarrow_{\mathcal{E}}^* \underbrace{f(t_1, \ldots)}_{\text{weil} \leftrightarrow_{\mathcal{E}}^* \text{Kongruenzrelation}}$

β: Wir wollen, dass $I(x) = [x] \curvearrowright \beta(x) = [x]$
Strukturelle Induktion: $I(t) = [t]_{\leftrightarrow_{\mathcal{E}}^*}$
Schritt(2): Zeige: $A = (\mathcal{A}, \alpha_f)$ ist Modell von $\mathcal{E}$, d.h. für alle Variablenbelegungen γ gilt für $J = (\mathcal{A}, \alpha, \gamma)$, dass $J \vDash \mathcal{E}$
Sei $u \equiv v \in \mathcal{E}$. Zu Zeigen: $J \vDash u \equiv v$
Für $x \in \mathcal{V}$ ist $\gamma(x) \in \mathcal{T}(\Sigma, \mathcal{V})/_{\leftrightarrow_{\mathcal{E}}^*}$, d.h. $\gamma(x) = [s_x]_{\leftrightarrow_{\mathcal{E}}^*}$ für einen Term s_x
Sei σ die Substitution $\{x/s_x|^{\mathcal{E}}$ für alle $x \in \mathcal{V}$, die in $\mathcal{E}$ auftreten$\}$
Dann gilt: $J(x) = \gamma(x) = [s_x]_{\leftrightarrow_{\mathcal{E}}^*} = [x\sigma]_{\leftrightarrow_{\mathcal{E}}^*}$
(Beispiel: $\mathcal{E}_{plus}$ zu Zeigen: $J \vDash plus(\mathcal{O}, x) \equiv Y$. $\gamma(y) = [s(\mathcal{O})]_{\leftrightarrow_{\mathcal{E}}^*}$
$s_y = succ(\mathcal{O})$
$\sigma = \{y/succ(\mathcal{O}), \}$
$J(y) = [y\sigma]_{\leftrightarrow_{\mathcal{E}}^*} = [succ(\mathcal{O})]_{\leftrightarrow_{\mathcal{E}}^*}$
$J(t) = [t\sigma]_{\leftrightarrow_{\mathcal{E}}^*}$

$J(plus(\mathcal{O}, y)\sigma) = [plus(\mathcal{O}, y)\sigma] = [plus(\mathcal{O}, succ(\mathcal{O}))]$

Durch strukturelle Induktion zeigt man $J(t) = [t\sigma]_{\leftrightarrow^*_\mathcal{E}}$
zu Zeigen war:
$J \vDash u \equiv v$, d.h. $J(u) = J(v)$, d.h. $[u\sigma]_{\leftrightarrow^*_\mathcal{E}} = [v\sigma]_{\leftrightarrow^*_\mathcal{E}}$
d.h. $u\sigma \leftrightarrow^*_\mathcal{E} v\sigma$ (Dies folgt aus $u \leftrightarrow_\mathcal{E} v$ und Stabilität von $\leftrightarrow_\mathcal{E}$, Lemma 3.1.13)
$\square$ (endlich)

Korrektheit: Wenn wir eine Gleichung beweisen durch Herleitung ($s \leftrightarrow^*_\mathcal{E} t$), dann ist sie wirklich wahr.
Vollständigkeit: Auf diese Weise kann man jede wahre Gleichung beweisen.
Vorteil: Man kann $s \equiv_\mathcal{E} t$ ohne Rückgriff auf die Semantik systematisch / automatisch beweisen:
$s \leftrightarrow^*_\mathcal{E} t$ ist semi-entscheidbar (d.h: es exisitert ein autmatisches Verfahren, dass $s \leftrightarrow^*_\mathcal{E} t$ beweisen kann, wenn es gilt. Wenn $s \nleftrightarrow^*_\mathcal{E} t$, dann terminiert das Verfahren eventuell nicht)
Da $\leftrightarrow^*_\mathcal{E} \quad = \quad \equiv_\mathcal{E}$ ist auch das Wortproblem semi-entscheidbar.
Arbeitsweise des Semi-Entscheidbaren-Verfahrens:



Wurzel: mit s markiert
Kinder: Terme u mit $s \leftrightarrow_\mathcal{E} u$
Falls $s \leftrightarrow^*_\mathcal{E} t$, dann tritt irgendwann t im Baum auf $\curvearrowright$ Abbruch mit Erfolg
Sonst terminiert das Verfahren nicht.

Problem: Wenn $t_1 \equiv t_2 \in \mathcal{E}$ mit $\mathcal{V}(t_1) \neq \mathcal{V}(t_1)$, dann existieren unendlich viele u mit $s \leftrightarrow_\mathcal{E} u$ Aber auch dann lässt sich jeder Knoten konstruieren $\curvearrowright$ bleibt semi-entscheidbar [14.05.04]
Satz von Birkhoff: $s \equiv_\mathcal{E} t$ gdw. $s \leftrightarrow^*_\mathcal{E} t$
Überprüfung von $s \equiv_\mathcal{E} t$ durch $s \leftrightarrow^*_\mathcal{E} t$: automatisierbar, aber:

- Semi-Entscheidungsverfahren: ungeeignet, falls $s \not\equiv_\mathcal{E} t$ (Verfahren terminiert dann nicht)
- ineffizient (zuviel Wahlmöglichkeiten: welche Gleichung in welcher Richtung auf welchen Teilterm anwenden?)

Bsp: $i(i(n)) \equiv n$ Suchbaum der Tiefe 3: > 20 Knoten
Suchbaum der Tiefe 15: Falls jeder Knoten 3 Kinder hat: 3^{15} Knoten $>$ 14 Millionen
im folgenden: verbesserte Verfahren, die auf dem Satz von Birkhoff beruhen, aber zielgerichteter vorgehen.
3.2 Spezialfall: Keine Variablen ⇒ Kongruenzabschluss
3.3 Allgemeiner Fall (mit Variablen) ⇒ Termersetzung

3.2 Der Kongruenzabschluss bei Grundidentitäten

Spezialfall: $\mathcal{E}, s, t$ enthalten keine Variablen
Anwendungen:

- Compilerbau: Optimierungen des zu compilierenden Programms
  Bsp: common subexpression Problem
  Mehrfach auftretende $\underbrace{gleiche}$ Teilausdrücke sollten nur einmal ausgewertet werden

müssen nicht syntaktisch gleich sein, sondern es reicht, wenn aus dem davorstehenden Programmcode folgt, das die Ausdrücke gleich sind

- Programmverifikation: Kongruenzabschluss lässt sich verwenden, um ein Entscheidungsverfahren für Allgemeingültigkeit universeller prädikantenlogischer Aussagen zu erhalten.
- Kombination von Entscheidungsverfahren

Definition 3.2.1: Grundidentität
Eine Termgleichung $s \equiv t$ ist eine Grundidentität, falls sie keine Variablen enthält, d.h. $\mathcal{V}(s) = \mathcal{V}(t) = \emptyset$

Beispiel 3.2.2:
Variablen für Zahlen: i, j , k, l, m
Variablen für Arrays: f, g

```
i = j;
k = l;
f[i] = g[k];
if j==f[j] then
   m=g[l];
   .
   . ← (*)
   .
end if
```

An der Stelle (*) tritt Ausdruck auf, der sowohl $f[m]$ als auch $g[k]$ enthält. gilt an der Stelle (*), dass $f[m] = g[k]$?
Obige Frage lässt sich wie folgt umformulieren:

$\mathcal{E} = \{i \equiv j, k \equiv l, f(i) \equiv g(k), j \equiv f(j), m \equiv g(l)\}$: Gilt: $f(m) \equiv_{\mathcal{E}} g(k)$

- Jetzt sind i, j, k, l, m Konstanten, keine Variablen. Sonst würde „$i \equiv j$“ bedeuten, das „$\forall i, j \quad i \equiv j$“, d.h. alle Terme wären “gleich“. $\Rightarrow$ Sind wirklich Grundidentitäten.

- Ablesen der Gleichungen $\mathcal{E}$ aus dem Programm benötigt im Allgemeinen Programmanalyse aufgrund von Seiteneffekten

  i = j $\quad \mathcal{E} = \{i \equiv j, j \equiv j + 1\}$ ist falsch, beschreibt nicht den Zustand an Stelle (*)

  j = j+1 Richtig wäre $\mathcal{E} = \{i = j - 1\}$ (*)

Wie untersucht man $f(m) \equiv_{\mathcal{E}} g(k)$?
1. Möglichkeit: Versuche, $f(m) \leftrightarrow^*_{\mathcal{E}} g(k)$

- ineffizient, nicht tolerierbar, es existiert ein Entscheidungsverfahren mit polynomiellen Aufwand

- Semi-entscheidungsverfahren. Hier nicht tolerierbar, denn bei Grundidentitäten ist das Wortproblem entscheidbar!

Wesentliche Eigenschaften von $\equiv_{\mathcal{E}}$: stabil, monoton/kongruent,
$\underbrace{\text{reflexiv, symmetrisch, transitiv}}_{\text{Äquivalenzrelation}}$
Bei Grundidentitäten gibt es keine Variablen $\curvearrowright$ Stabilität uninteressant
Idee: Starte mit den Gleichungen aus $\mathcal{E}$ und berechne alle Gleichungen, die aus $\mathcal{E}$ aufgrund der direkten Anwendung von Reflexivität, Symmetrie, Transitivität und Kongruenz folgen.

Definition 3.2.3: Direkte Anwendung von Reflexivität, Symmetrie, Transitivität, Kongruenz
Für jede Menge $\mathcal{E}$ von Grundidentitäten definieren wir:

- $R = \{t \equiv t | t \in \mathcal{T}(\mathcal{E})\}$
- $S(\mathcal{E}) = \{t \equiv s | s \equiv t \in \mathcal{E}\}$
- $T(\mathcal{E}) = \{s \equiv v |$ es existiert ein $t \in \mathcal{T}(\Sigma)$ mit $s \equiv t$ und $t \equiv v \in \mathcal{E}\}$
- $C(\mathcal{E}) = \{f(s_1, \ldots, s_n) \equiv f(t_1, \ldots, t_n) | f \in \Sigma, s_i \equiv t_i \in \mathcal{E}$ für alle $1 \leq i \leq n\}$

Beispiel 3.2.4:
Sei $\mathcal{E}$ wie in Beispiel 3.2.2, $\mathcal{E} = \underbrace{i, j, k, l, m}_{\Sigma_0}, \underbrace{f, g}_{\Sigma_1}$:
$R = \{i \equiv i, j \equiv j, f(i) \equiv f(i), f(f(i)) \equiv f(f(i)), \ldots\}$
$S = \{j \equiv i, l \equiv k, g(k) \equiv f(i), f(j) \equiv j, g(l) \equiv m\}$

$T(\mathcal{E}) = \{i \equiv f(j)\}$
$C(\mathcal{E}) = \{f(i) \equiv f(j), g(i) \equiv g(j), \ldots\}$ (10 Gleichungen)
In $\mathcal{E} \cup R \cup S(\mathcal{E}) \cup T(\mathcal{E}) \cup C(\mathcal{E})$ stehen nur Gleichungen, die aus $\mathcal{E}$ folgen. um $s \equiv_{\mathcal{E}} t$ zu überprüfen, könnte man testen, ob $s \equiv t \in \mathcal{E} \cup R \cup S(\mathcal{E}) \cup T(\mathcal{E}) \cup C(\mathcal{E})$.
Aber aus $s \equiv t \notin \underbrace{\mathcal{E} \cup R \cup S(\mathcal{E}) \cup T(\mathcal{E}) \cup C(\mathcal{E})}_{\text{enthält nicht alle Gleichungen, die aus } \mathcal{E} \text{ folgen}}$ folgt nicht $s \not\equiv_{\mathcal{E}} t$.
Problem: Symmetrie, Transitivität, Kongruenz müssten mehrfach angewendet werden.

Definition 3.2.5: Kongruenzabschluss
Für eine Menge von Grundidentitäten $\mathcal{E}$ definieren wir die Mengen $\mathcal{E}_0, \mathcal{E}_1, \mathcal{E}_2, \ldots$ wie folgt:

- $\mathcal{E}_0 = \mathcal{E} \cup R$
- $\mathcal{E}_{i+1} = \mathcal{E}_i \cup S(\mathcal{E}_i) \cup T(\mathcal{E}_i) \cup C(\mathcal{E}_i)$

Der Kongruenzabschluss $CC(\mathcal{E})$ ("congruence closure") ist der "Limes" der FOlge $\mathcal{E}_0, \mathcal{E}_1, \ldots$, d.h. $CC(\mathcal{E}) = \cup_{i \in \mathbb{N}} \mathcal{E}_i$

Beispiel 3.2.6:
Wir zeigen $f(m) \equiv g(k) \in CC(\mathcal{E})$, denn $f(m) \equiv g(k) \in \mathcal{E}_g$

oberste Zeile: Gleichungen aus $\mathcal{E}_0$
nächste Zeile: Gleichungen aus $\mathcal{E}_1$
letzte Zeile: Gleichungen aus $\mathcal{E}_g$
} nur die Gleichungen aus $\mathcal{E}_i$ ausgeben, die für den Beweis von $f(m) = g(k)$ wichtig sind

Satz 3.2.7: Kongruenzabschluss ist korrekt und vollständig
Sei $\mathcal{E}$ eine Mengen von Grundindentitäten über Σ und sei $s, t \in \mathcal{T}(\Sigma)$. Dann gilt $s \equiv_{\mathcal{E}} t$, gdw. $s \equiv_{\mathcal{E}} t \in CC(\mathcal{E})$
Beweis: Korrektheit: $s \equiv t \in \underbrace{CC(\mathcal{E})}_{\mathcal{E}_0 \cup \mathcal{E}_1 \cup \ldots} \curvearrowright s \equiv_{\mathcal{E}} t$ d.h. falls $s \equiv t \in \mathcal{E}_i \curvearrowright s \equiv_{\mathcal{E}} t$
Induktion über i
Induktionsanfang i=0
$\mathcal{E}_0 = \mathcal{E} \cup R$, denn $\equiv_{\mathcal{E}}$ ist reflexiv
Induktionsschluss $i \curvearrowright i + 1$
$s \equiv t \in \mathcal{E}_{i+1} = \mathcal{E}_i \cup S(\mathcal{E}_i) \cup T(\mathcal{E}_i) \cup C(\mathcal{E}_i)$
Induktionshypothese: für alle $u \equiv v \in \mathcal{E}_i$ gilt: $u \equiv_{\mathcal{E}} v$. Damit folgt $s \equiv_{\mathcal{E}} t$, da $\equiv_{\mathcal{E}}$ ~~reflexiv~~, symmetrisch, transitiv, Kongruent (Lemma 3.1.10)
Vollständigkeit $s \equiv_{\mathcal{E}} t \curvearrowright s \equiv t \in CC(\mathcal{E})$
Wir zeigen später (Satz 3.2.12) eine stärkere Aussage $\curvearrowright$ lasse Beweis hier weg
□
Alternative Methode (statt $s \leftrightarrow_{\mathcal{E}}^* t$) um Wortproblem bei Grundidentitäten zu lösen: Konstruiere $\mathcal{E}_0, \mathcal{E}_1, \ldots$ bis man ein $\mathcal{E}_i$ findet mit $s \equiv t \in \mathcal{E}_i$ ist immer noch ineffizient und Semi-Entscheidungsverfahren.

Um Entscheidungverfahren zu bekommen: stoppe Iteration $\mathcal{E}_0, \mathcal{E}_1, \ldots$ nach endlicher Zeit.
Im Allgemeinen gilt: $\mathcal{E}_i \subsetneq \mathcal{E}_{i+1}$ für alle i: $i \equiv j \in \mathcal{E}_0$
$g(i) \equiv g(j) \in \mathcal{E}_1 \backslash \mathcal{E}_0$
$g(g(i)) \equiv g(g(j)) \in \mathcal{E}_2 \backslash \mathcal{E}_1$
Die meisten dieser Gleichungen sind für den Beweis einer vorgegebenen Gleichung $s \equiv t$ unnötig!
Erkenntnis: Man muss nur Gleichungen $u \equiv v$ betrachten, bei denen u, v Teilterme aus $\mathcal{E}, s, t$ sind.

[18.05.04]

Ziel: $s \equiv_{\mathcal{E}} t$? Spezialfall: Keine Variablen
Problem bei Kongruenzabschluss: Wenn $s \not\equiv_{\mathcal{E}} t$, dann terminiert das Verfahren im Allgemeinen nicht.
Abhilfe: Konstruiere nicht mehr alle Terme in $\mathcal{E}_i$.

Definition 3.2.8: Teiltermmenge
Zu jedem Term s sei $Subterms(s) = \{s|_\pi | \pi \in Occ(s)\}$ die Menge seiner Teilterme. Zu einer Menge von Gleichungen $\mathcal{E}$ sei $Subterms(s) = \cup_{s \equiv t \in \mathcal{E}} Subterms(s) \cup Subterms(t)$

Wenn $\mathcal{E}$ endlich ist, ist auch $Subterms(\mathcal{E})$ endlich.
Bei der Bildung des Kongruenzabschlusses zum Nachweis von $s \equiv_{\mathcal{E}} t$ muss man nur Gleichungen zwischen Termen aus $S = Subterms(\mathcal{E}) \cup Subterms(s) \cup Subterms(t)$ berücksichtigen.

Definition 3.2.9: Kongruenzabschluss bezüglich einer Menge von Termen
Sei $\mathcal{E}$ eine Menge von Grundidentitäten, sein $s, t, \in \mathcal{T}(\Sigma)$. Sei $S = Subterms(\mathcal{E}) \cup Subterms(s) \cup Subterms(t)$
Wir definieren: $\mathcal{E}_0^S = (\mathcal{E} \cup R) \cap S \times S$
$\leftarrow$ Hier sehen wir die Gleichungen als Paare von Termen an (bedeutet $\mathcal{E}_0^S = \{u \equiv v \in \mathcal{E} \cup R | u \in S, v \in S\}$
$\mathcal{E}_{i+1}^S = (\mathcal{E}_i^S \cup S(\mathcal{E}_i^S) \cup T(\mathcal{E}_i^S) \cup C(\mathcal{E}_i^S)) \cap S \times S$

(Wenn $|S| = n$, dann $|\mathcal{E}_i^S| \leq n^2$. Nach spätestens n^2 Schritten stoppt die Iteration)
Der Kongruenzabschluss bezüglich S ist $CC^S(\mathcal{E}) = \cup_{i \in \mathbb{N}} \mathcal{E}_i^S$

Beispiel 3.2.10: Kongruenzabschluss bezüglich S für $\mathcal{E}$ aus Beispiel 3.2.2
$\mathcal{E}_0 = \mathcal{E} \cup \{t \equiv t | t \in \mathcal{T}(\Sigma)\}$
$\mathcal{E}_0^S = \mathcal{E} \cup \{t \equiv t | t \in S\}$ (5 Gleichungen)

$$\mathcal{E}_1 = \mathcal{E}_0 \cup \{j \equiv i, \ldots, g(l) \equiv m\} \cup \{i \equiv f(j)\} \cup \underbrace{\{f(i) \equiv f(j), g(i) \equiv g(j), \ldots\}}_{\text{10 Gleichungen}}$$

$$\mathcal{E}_i^S = \mathcal{E}_0^S \cup \{j \equiv i, \ldots, g(l) \equiv m\} \cup \{i \equiv f(j)\} \cup \underbrace{\{f(i) \equiv f(j), g(k) \equiv g(l)\}}_{\text{2 Gleichungen}}$$

Lemma 3.2.11: $CC^S(\mathcal{E})$ wird nach endlich vielen Schritten erreicht
Seien $\mathcal{E}, s, t$ wie in Definition 3.2.9, $\mathcal{E}$ endlich. 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: Für alle $i \in \mathbb{N}$: $\mathcal{E}_i^S \subseteq \mathcal{E}_{i+1}^S$. ($|\mathcal{E}_i^S| = n^2$ Wenn alle Gleichungen in $\mathcal{E}$ "gleich" sind)! Da $\mathcal{E}_i^S \subseteq S \times S$ und da $S \times S$ endlich ist, muss es ein $i \in \mathbb{N}$ geben mit $\mathcal{E}_i^S = \mathcal{E}_{i+1}^S$. Dann gilt $\mathcal{E}_i^S = \mathcal{E}_j^S$ für $j \geq i$. Somit: $CC^S(\mathcal{E}) = \cup_{j \in \mathbb{N}} \mathcal{E}_j^S = \mathcal{E}_i^S$ □

Satz 3.2.12: Kongruenzabschluss bezüglich S ist korrekt und vollständig
Sei $\mathcal{E}$ eine Menge von Grundidentitäten, $s, t, \in \mathcal{T}(\Sigma)$, Sei $Subterms(\mathcal{E}) \cup Subterms(s) \cup Subterms(t) \subseteq S$.
Dann gilt: $s \equiv_{\mathcal{E}} t$ gdw. $s \equiv_{\mathcal{E}} t \in CC^S(\mathcal{E})$

Beweis: Korrektheit: $s \equiv t \in CC^S(\mathcal{E}) \subseteq CC(\mathcal{E}) \overset{\text{Satz 3.2.7}}{\curvearrowright}{}^{1} s \equiv_{\mathcal{E}} t$
Vollständigkeit: zu Zeigen: $s \equiv_{\mathcal{E}} t \curvearrowright s \equiv t \in CC^S(\mathcal{E})$
Satz von Birkhoff (Satz 3.1.14): $s \equiv_{\mathcal{E}} t \curvearrowleft s \leftrightarrow_{\mathcal{E}}^* t$.
Es reicht also zu Zeigen: $s \leftrightarrow_{\mathcal{E}}^* t \curvearrowright s \equiv t \in CC^S(\mathcal{E})$
Vorraussetzung: $s \leftrightarrow_{\mathcal{E}}^n t$ Induktion über n (Länge der Herleitung)
Induktionsanfang: $n = 0 \curvearrowright s = t \curvearrowright s \equiv t \in R \cap (S \times S) \subseteq \mathcal{E}_0^S \subseteq CC^S(\mathcal{E})$.
Induktionsschluss: $n > 0 : s = s_0 \leftrightarrow_{\mathcal{E}} s_1 \leftrightarrow_{\mathcal{E}} \ldots \leftrightarrow_{\mathcal{E}} s_n = t$

Fall 1 Mindestens einmal wurde eine Gleichung aus $\mathcal{E}$ an der obersten Stelle $\mathcal{E}$ ausgewertet.

Es existiert k mit $1 \leq k < n$ mit $s_k = u$, $s_{k+1} = v$ und $u \equiv v$ oder $v \equiv u \in \mathcal{E}$.

$s = s_0 \leftrightarrow_{\mathcal{E}} \ldots \leftrightarrow_{\mathcal{E}} s_k = u, v = s_{k+1} \leftrightarrow_{\mathcal{E}} \ldots \leftrightarrow_{\mathcal{E}} s_n = t$. Beide Herleitungen $s \leftrightarrow_{\mathcal{E}}^* u, v \leftrightarrow_{\mathcal{E}}^* t$ sind kürzer als $s \leftrightarrow_{\mathcal{E}}^* t$

Es gilt $Subterms(u) \subseteq S, Subterms(v) \subseteq S$, denn $u \equiv v$ oder $v \equiv u \in \mathcal{E}$

Induktionshypothese: $s \equiv u \in CC^S(\mathcal{E})$, $v \equiv t \in CC^S(\mathcal{E})$

$\curvearrowright$ es existiert $i \in \mathbb{N}$: $s \equiv u \in \mathcal{E}_i^S, v \equiv t \in \mathcal{E}^S, u \equiv v \in \mathcal{E}_1^S \subseteq \mathcal{E}_i^S$

$\curvearrowright s \equiv v \in \mathcal{E}_{i+1}^S \curvearrowright s \equiv t \in \mathcal{E}_{i+2}^S \subseteq CC^S(\mathcal{E})$

Fall 2 In der Herleitung $S \leftrightarrow_{\mathcal{E}}^* t$ wurde nie eine Gleichung ganz aussen angewendet.

[Grafik] Seien $\pi_1, \ldots, \pi_k$ mit $\pi_i \perp \pi_j$ für $i \neq j$, die obersten Stellen in S, an denen in der Herleitung jemals Gleichungen angewendet werden.

$\curvearrowright s|_{\pi_i} = p_i$, $p_i \leftrightarrow_{\mathcal{E}}^* q_i$, $t = s[q_1]_{\pi_1} \ldots [q_k]_{\pi_k}$

Länge der Herleitung $P_i \leftrightarrow_{\mathcal{E}}^* q_i$ ist höchstens n. In jeder dieser Herleitungen wird aber mindestens einmal eine Gleichung aus $\mathcal{E}$ ganz aussen angewendet
$\overset{\text{Fall 1}}{\curvearrowright}$ $p_i \equiv q_i \in CC^S(\mathcal{E})$

Da $CC^S(\mathcal{E})$ unter Kongruenz abgeschlossen ist (auf Termen von S) folgt auch $s \equiv t \in CC^S(\mathcal{E})$

[1] Korrektheit von $CC(\mathcal{E})$

□

Aus der Vollständigkeit von $CC^S(\mathcal{E})$ folgt Vollständigkeit von $CC(\mathcal{E})$ (Satz 3.2.7): $s \equiv_{\mathcal{E}} t \curvearrowright s \equiv t \in CC^S(\mathcal{E}) \subseteq CC(\mathcal{E})$

Beispiel 3.2.13:
Im Beweis von $f(m) \equiv_{\mathcal{E}} g(k)$ erkennt man: es traten nur Gleichungen aus $S \times S$ auf: $f(m) \equiv g(k) \in \mathcal{E}_g^S$.

Korollar 3.2.14:
Sei $\mathcal{E}$ eine endliche Menge von Grundidentitäten. Dann ist das Wortproblem (d.h.: „$s \equiv_{\mathcal{E}} t$“) entscheidbar.

Praktische Berechnung: Bilde nicht alle Gleichungen in $\mathcal{E}_i^S$, sondern repräsentiere Äquivalenzklassen.

Definition 3.2.15:
$\leftrightarrow_{\mathcal{E}}$ als $u \leftrightarrow_{\mathcal{E}} v$ gdw. $u \equiv v \in \mathcal{E}$

Bilde reflexiv-symmetrisch-transitive Hülle von $\leftrightarrow_{\mathcal{E}}$: $\leftrightarrow_{\mathcal{E}}^*$. $S_0 = \{[s]_{\leftrightarrow_{\mathcal{E}}^*} | s \in S\}$
Starte mit $\{\{s,t\} | s \equiv t \in \mathcal{E}\} \cup \{\{s\} | s \in S\}$ und vereinige danach Mengen, die nicht disjunkt sind.
Bilde zusätzlich $f(s1, \ldots, s_n), f(t_1, \ldots, t_n)$ falls s, t zusammen in einer Äquivalenzklasse auftreten und $f(s_1, \ldots, s_n), f(t_1, \ldots, t_n) \in \mathcal{E}$.
Wende abwechselnd Kongruenz und Äquivalenz an. [21.05.04]
Algorithmus für Kongruenzabschluss: Konstruiere nicht alle Gleichungen in $CC^S(\mathcal{E})$, sondern konstruiere nur Äquivalenzklassen.
$s \leftrightarrow_{\mathcal{E}} t$ gdw. $s \equiv t \in \mathcal{E}$

$$S_0 = \{[S]_{\leftrightarrow_{\mathcal{E}}^*} | s \in S\} \qquad \mathcal{E}_0^{s'} = \{s \equiv t | \text{ es existiert } M \in S_0 \text{ mit } s, t \underbrace{\in}_{\text{d.h. } s \leftrightarrow_{\mathcal{E}}^* t} M\}$$

$$S_1 = \{[S]_{\leftrightarrow_0^{S'} \cup C(\mathcal{E}_0^S)'} | s \in S\} \quad \mathcal{E}_1^{s'} = \{s \equiv t | \text{ es existiert } M \in S_1 \text{ mit } s, t \underbrace{\in}_{\text{d.h. } s \leftrightarrow_{\mathcal{E}}^* t} M\}$$

$$S_2 = \{[S]_{\leftrightarrow_0^{S'} \cup C(\mathcal{E}_1^S)'} | s \in S\} \quad \mathcal{E}_2^{s'} = \{s \equiv t | \text{ es existiert } M \in S_2 \text{ mit } s, t \underbrace{\in}_{\text{d.h. } s \leftrightarrow_{\mathcal{E}}^* t} M\}$$

Algorithmus: Kongruenzabschluss $(\mathcal{E}, s, t)$
Eingabe: endliche Menge von Grundidentitäten $\mathcal{E}$, zwei Grundterme s, t
Ausgabe: „True“, falls $s \leftrightarrow_{\mathcal{E}}^* t$, sonst „False“

1. Sei $S = Subterms(\mathcal{E}) \cup Subterms(s) \cup Subterms(t)$

2. Sei $L = \{\{s,t\} | s \equiv t \in \mathcal{E}\} \cup \{\{s\} | s \in S\}$

„Äq“ 3. Vereinige alle Mengen $M_1, M_2 \in L$ mit $M_1 \cap M_2 \neq \emptyset$. (danach gilt: $L = S_0$)

„Kon“ 4. Sei $K = L \cup \{\{f(s_1, \ldots, s_n), f(t_1, \ldots, t_n)\} | f \in \Sigma$ es existiert $M_i \in L$ mit $s_i, t_i \in M_i$(für alle$1 \leq i \leq n), f(s_1, \ldots, s_n), f(t_1, .., t_n) \in S\}$

„Äq“ 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. (Hier gilt $L = S_i$ im i-ten Schleifendurchlauf)

7. Falls es ein $M \in L$ gibt mit $s, t \in M$, dann gib „True“ aus, sonst gib „False“ aus.

Kongruenzabschluss beruht auf der Idee, dass man zum Beweis von $s \leftrightarrow_{\mathcal{E}}^* t$ nur Teilterme von $\mathcal{E}, s, t$ betrachten muss. Stimmt nicht mehr, wenn Variablen auftreten:
$\mathcal{E} = \{f(f(x)) \equiv g(x)\}$ Gilt $f(g(a)) \equiv_{\mathcal{E}} g(f(a))$?
$f(g(a)) \leftarrow_{\mathcal{E}} f(f(f(a))) \rightarrow_{\mathcal{E}} g(f(a))$
Herleitung gelingt nur, wenn zwischendurch Terme wie $f^3(a)$ betrachtet werden, die keine Teilterme $\mathcal{E}, s, t$ sind.

3.3 Termersetzungssysteme

Satz von Birkhoff: Statt $s \equiv_{\mathcal{E}} t$ kann man $s \leftrightarrow_{\mathcal{E}}^* t$ zeigen. Nachteile: ineffizient und falls $s \not\equiv_{\mathcal{E}} t$, dann terminiert das Verfahren nicht.
Erhöhung der Effizienz: verringere Indeterminismen des obigen Verfahrens:

1. in welche Richtung soll eine Gleichung angewendet werden?
2. welche Gleichung soll auf welchen Teilterm angewendet werden?
3. wie lange sollte man Gleichungen anwenden?

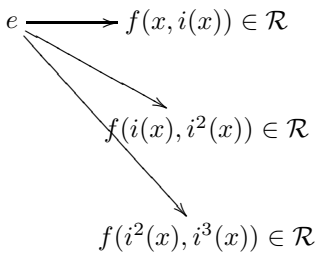
Lösung von (1): Gleichungen nur noch von links nach rechts anwenden. Schreibweise für gerichtete Gleichung: ersetze „$\equiv$“ durch „$\rightarrow$“. Eine gerichtete Gleichung nennt man Regel, eine Menge von gerichteten Gleichungen nennt man Termersetzungssystem

Definition 3.3.1: Termersetzungssystem
Für $l, r \in \mathcal{T}(\Sigma, \mathcal{V})$ heisst $l \leftarrow r$ Regel, falls

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

Eine Menge $\mathcal{R}$ von Regeln heisst Termersetzungssystem (TES, term rewriting system, TRS)
Für ein TES $\mathcal{R}$ ist Ersetzungsrelation $\rightarrow_{\mathcal{R}} \subseteq \mathcal{T}(\Sigma, \mathcal{V}) \times \mathcal{T}(\Sigma, \mathcal{V})$ definiert als $s \rightarrow_{\mathcal{R}} t$ gdw. $s|_p i = l\sigma$ und $t = s[r\sigma]_\pi$ für $\pi \in Occ(s)$, $l \rightarrow r \in \mathcal{R}$, $\sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})$. Man nennt $s \rightarrow_{\mathcal{R}} t$ „Termersetzungsschritt“, bei dem s an der Stelle π reduziert wird.
Ein Teilterm $s|_\pi$, auf den die linke Seite einer Regel matcht, heisst Redex („Reducible expression“)
Schreibe auch „$s \rightarrow t$“ statt „$s \rightarrow_{\mathcal{R}} t$“, wenn $\mathcal{R}$ klar ist.
Schreibe auch „$t \leftarrow s$“ statt „$s \rightarrow_{\mathcal{R}} t$“, wenn $\mathcal{R}$ klar ist.

Grund für $\mathcal{V}(r) \subseteq \mathcal{V}(l)$: Für jeden Term s existieren nur endlich viele $\underbrace{t}$ mit
Wie bei Gleichungssystemen beschränken wir uns auf endliche TESe.
$s \rightarrow_{\mathcal{R}} t$
sonst: $e \rightarrow f(x, i(x)) \in \mathcal{R}$:



$\mathcal{V}(r) \subseteq \mathcal{V}(l)$ garantiert, dass der Matcher σ eindeutig festliegt, wenn $s|_\pi = l\sigma$. Hierdurch liegt dann auch $r\sigma$ fest.
Weiterer Grund: sonst würde Anwendung von Regeln nie terminieren
Grund für $l \notin \mathcal{V}$: Reduziere Indeterminismen ($x \rightarrow t$ wäre immer / auf jeden Term anwendbar) und Terminierung

Beispiel 3.3.2:
Ziel: Löse Wortproblem für Gleichungssystem $\mathcal{E}$
Vorgehen: Überführe $\mathcal{E}$ in ein TES $\mathcal{R}$
$plus(\mathcal{O}, y) \equiv y$
$plus(\mathcal{O}, y) \rightarrow y$ (Richtung $y \rightarrow plus(\mathcal{O}, y)$ verletzt Variablen-bedingung)
$plus(succ(x), y) \equiv succ(plus(x, y))$
$plus(succ(x), y) \rightarrow succ(plus(x, y))$ (Andere Richtung wäre auch möglich)
Um von $\mathcal{E}$ zu $\mathcal{R}$ zu kommen, ist es nicht immer ratsam, eine Gleichung $s \equiv t$ durch $s \rightarrow t$ oder $t \rightarrow s$ zu ersetzen.
Grund: Um Indeterminismen (2) und (3) zu lösen, möchte man TESe $\mathcal{R}$ konstruieren, bei denen $\rightarrow_{\mathcal{R}}$-Anwendung terminiert und eindeutige Ergebnisse liefert.

Damit sich $\mathcal{E}$ und $\mathcal{R}$ „entsprechen“, müssen sie „äquivalent“ sein:

Definition 3.3.3: Äquivalenz von Gleichungssystem und TES
Sei $\mathcal{E}$ ein Gleichungssystem, sei $\mathcal{R}$ ein TES über Σ und $\mathcal{V}$. Dann ist $\mathcal{R}$ äquivalent zu $\mathcal{E}$ gdw. $\leftrightarrow^*_{\mathcal{R}} = \leftrightarrow^*_{\mathcal{E}}$

Falls $\mathcal{E}$ und $\mathcal{R}$ äquivalent sind, dann $s \equiv_\mathcal{E} t \underbrace{\text{gdw.}}_{\text{Birkhoff}} s \leftrightarrow^*_\mathcal{E} t \underbrace{\text{gdw.}}_{\text{Äquivalenz}} s \leftrightarrow^*_\mathcal{R} t$.
Also: Durch Anwendung von Regeln aus $\mathcal{R}$ (in beide Richtungen) kann man genau alle wahren Aussagen beweisen. Wenn $\mathcal{R}$ und $\mathcal{E}$ gegeben, wie kann man überprüfen, ob sie äquivalent sind?

Satz 3.3.4: Zusammenhang zwischen Gleichungssystem und TES
Sei $\mathcal{E}$ ein Gleichungssystem, Sei $\mathcal{R}$ ein TES über Σ, $\mathcal{V}$. $\mathcal{R}$ ist äquivalent zu $\mathcal{E}$ gdw.

- $l \leftrightarrow^*_\mathcal{E} r$ 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 $\mathcal{E}$“)

Beweis: Übung. (Äquivalenz $\curvearrowright$ Korrektheit und Adäquanz (trivial). „$\curvearrowleft$“ folgt aus Monotonie und Stabilität von $\leftrightarrow^*_\mathcal{E}, \leftrightarrow^*_\mathcal{R}$ Induktion über die Anzahl der Herleitungsschritte).
□

Falls $\mathcal{R}$ aus $\mathcal{E}$ entsteht, indem „$s \equiv t$“ durch „$s \to t$“, oder „$t \to s$“ ersetzt wird, so sind $\mathcal{R}$ und $\mathcal{E}$ trivialerweise äquivalent.

Beispiel 3.3.5:
$\mathcal{E} = \{b \equiv c, b \equiv a, f(a) \equiv f(f(a))\}$ $\mathcal{R} = \{c \to b, a \to b, f(f(b)) \to f(b)\}$

$\mathcal{R}$ ist korrekt für $\mathcal{E}$: $c \leftarrow_\mathcal{E} b$
$a \leftarrow_\mathcal{E} b$
$f(f(b)) \to_\mathcal{E} (f(f(a))) \leftarrow_\mathcal{E} f(a) \leftarrow_\mathcal{E} f(b)$

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

$\mathcal{R}' = \{c \to a, f(f(b)) \to f(b)\}$ korrekt, aber nicht adäquat $\curvearrowright$ nicht äquivalent

[25.05.04]

Wortproblem: $s \equiv_\mathcal{E} t$
Vorgehen: Suche TES $\mathcal{R}$, das zu $\mathcal{E}$ äquivalent ist
Birkhoff: $s \equiv_\mathcal{E} t$ gdw. $s \leftrightarrow^*_\mathcal{E} t$ gdw. $s \leftrightarrow^*_\mathcal{R} t$
Um TESe effizient und als Entscheidungsverfahren anwenden zu köennen, wollen wir wie folgt vorgehen, um $s \equiv_\mathcal{E} t$ zu überprüfen.

- Werte s und t solange wie möglich aus, d.h. wende TES-Regeln von links nach rechts an solange möglich.

  $s \to_\mathcal{R} s_1 \to_\mathcal{R} s_2 \to_\mathcal{R} \ldots \to_\mathcal{R} s_n$

  $t \to_\mathcal{R} t_1 \to_\mathcal{R} \ldots \to_\mathcal{R} t_n$

- Überprüfe, ob s_n und t_n syntaktisch gleich sind. Wenn ja, gib „True“ aus, wenn nein, gib „False“ aus.

Beispiel 3.3.6:
$\mathcal{E}$ = plus-Gleichungen, $\mathcal{R}$ = TES auf Folie (äquivalent zu $\mathcal{E}$)
untersuche: $plus(succ(succ(\mathcal{O})), x) \equiv_{\mathcal{E}} plus(succ(\mathcal{O}), succ(x))$?
$plus(succ(succ(\mathcal{O})), x) \to_{\mathcal{R}} succ(plus(succ(\mathcal{O}), x)) \to_{\mathcal{R}} succ(succ(plus(\mathcal{O}, x))) \to_{\mathcal{R}}$
$succ(succ(x))$
$plus(succ(\mathcal{O}), succ(x)) \to_{\mathcal{R}} succ(plus(\mathcal{O}, succ(x))) \to_{\mathcal{R}} succ(succ(x))$
Sind syntaktisch gleich $\curvearrowright$ Gleichung folgt aus $\mathcal{E}$ $s \equiv t$
$plus(succ(\mathcal{O}), succ(x)) \to_{\mathcal{R}} succ(plus(\mathcal{O}, succ(x))) \to_{\mathcal{R}} succ(succ(x))$

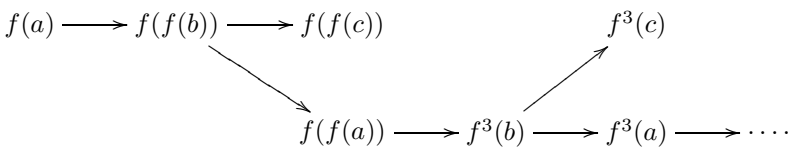
Probleme bei diesem Verfahren:

1. Mann kann s und t nicht zu s_n, t_n reduzieren, die selbst nicht mehr reduzierbar sind

2. s und t könnten zwar endliche Reduktionen, aber auch unendliche Reduktionen haben. Wie findet man die endlichen Reduktionen?

$\Rightarrow$ Einschränkung auf terminierende TESe, d.h. jede Reduktion $\ldots \to_{\mathcal{R}} \ldots \to_{\mathcal{R}} \ldots$ ist endlich.

Beispiel 3.3.7:
$\mathcal{E} = \{b \equiv c; b \equiv a, f(a) \equiv f(f(a))\}$
$\mathcal{R}_1 = \{b \to c, b \to a, f(a) \to f(f(a))\}$
Untersuche $f(a) \equiv_{\mathcal{E}} f(c)$?
$f(a) \to_{\mathcal{R}_1} f(f(a)) \to_{\mathcal{R}_1} f^3(a) \to_{\mathcal{R}_1} f^4(a) \to_{\mathcal{R}_1} \ldots$f(a) hat nur unendliche Auswertungen (1)
Definieren uns neues TES:
$\mathcal{R}_2 = \{b \to c, b \to a, f(a) \to f(f(b))\}$
Untersuche $f(a) \equiv_{\mathcal{E}} f(c)$?



f(a) hat sowohl endliche als auch unendliche Auswertungen (2).

Probleme: (3) Ziel: $s \leftrightarrow^*_{\mathcal{R}} t$ zeigen. Wir wollen aber nur untersuchen, ob es einen Term q mit $s \to^*_{\mathcal{R}} q \leftarrow^*_{\mathcal{R}} t$. Es könnte sein, das man zum Nachweis von $s \leftrightarrow^*_{\mathcal{E}} t$ die Richtung der Regelanwendung mehrmals ändern muss.
(4) Es könnte sein, das man bei geeigneter Regelanwendung $s \to^* q \leftarrow^* t$ zeigen könnte, bei ungeeigneter Regelanwendung aber nicht.
(Probleme (3) + (4): betreffen Eindeutigkeit)
$\Rightarrow$ Einschränkung auf konfluente TESe, d.h. jede maximale Reduktion liefert das gleiche Ergebnis.

Beispiel 3.3.7: (Fortsetzung)

$\mathcal{R}_3 = \{b \to c, b \to a, f(f(a)) \to f(a)\}$ terminiert
Untersuche $f(a) \equiv_\mathcal{E} f(c)$? Weder f(a) noch f(c) lassen sich reduzieren $\curvearrowright$ „False“
Aber: $f(a) \leftarrow_\mathcal{R} f(b) \rightarrow_\mathcal{R} f(c)$

Untersuche $f(f(b)) \equiv_\mathcal{E} f(a)$ mit $\mathcal{R}_3$:
f(a) nicht weiter reduzierbar

$$f(f(b)) \longrightarrow f(f(c)) \qquad \text{„False“}$$
$$\searrow$$
$$f(f(a)) \longrightarrow f(a) \qquad \text{„True“} \qquad (4)$$

In diesem Beispiel: keines der TESe, die aus $\mathcal{E}$ entstehen, in dem man $s \equiv t$ durch $s \to t$ oder $t \to s$ ersetzt, ist sowohl terminierend als auch konfluent. ein äquivalentes TES, das terminiert und konfluent ist, ist:
$\mathcal{R} = \{c \to b, a \to b, f(f(b)) \to f(b)\}$
$\Rightarrow$ Es kann sinnvoll sein, zu einem Gleichungssystem $\mathcal{E}$ ein äquivalentes TES $\mathcal{R}$ zu konstruieren, das nicht aus Gleichungen durch Ersetzung von „$\equiv$“ durch „$\rightarrow$“ oder „$\leftarrow$“ entsteht, denn dann ist $\mathcal{R}$ eventuell nicht terminierend bzw. nicht konfluent.
Fragen:

- Wie zeigt man die Terminierung eines TES? $\leftarrow$ Terminierung von Programmen (Kapitel 4)
- Wie zeigt man die Konfluenz eines TES? $\leftarrow$ Eindeutigkeit von Programmen (Kapitel 5)
- Wie bekommt man zu einem Gleichungssytsem $\mathcal{E}$ ein dazu äquivalentes TES Wie zeigt man die Terminierung eines TES $\mathcal{R}$, das terminiert und konfluent ist? $\leftarrow$ Vervollständigung von Programmen (Kapitel 6)

Vorher: Definiere „Terminierung“, „Konfluenz“ etc.
TES $\mathcal{R}$ terminiert gdw. es keine unendlichen Reduktionen $\ldots \rightarrow_\mathcal{R} \ldots \rightarrow_\mathcal{R} \ldots \rightarrow_\mathcal{R} \ldots \rightarrow_\mathcal{R} \ldots$ gibt

Definition 3.3.8: Fundierte Relation
Sei $\rightarrow$ eine Relation über einer Menge M. Die Relation ist fundiert (well-founded) gdw. es keine unendliche Folge $t_0, t_1, \ldots \in M$ gibt mit $t_0 \rightarrow t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow \ldots$

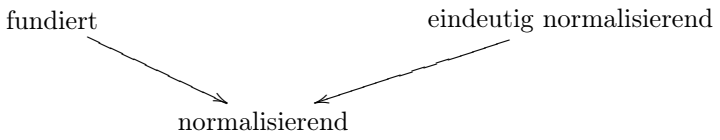
Beispiel 3.3.9:

- Die Relation $>_\mathbb{N}$ ist fundiert $(10 > 7 > 3 > 1 > 0)$
- Die echte Teiltermrelation $\rhd$ auf $\mathcal{T}(\Sigma, \mathcal{V})$ ist fundiert
  $(f(g(x), h(x)) \rhd g(x) \rhd x)$
- Die Relation $<_\mathbb{N}$ ist nicht fundiert $(0 < 1 < 2 < 3 \ldots)$

- Die Relation $>_{\mathbb{Z}}$ ist nicht fundiert $(2 > 1 > 0 > -1 > -2 > \ldots)$
- Die Relation $>_{\mathbb{Q}^+}$ ist nicht fundiert $(1 > \frac{1}{2} > \frac{1}{3} > \frac{1}{4} > \ldots)$
- Die Relation $>_{\mathbb{N}^*}$ ($\pi_1 >_{\mathbb{N}^*} \pi_2$ gdw. $\pi_1 = \pi_2\ \pi$ für ein $\pi \neq \mathcal{E}$) ist fundiert $31240 >_{\mathbb{N}^*} 312 >_{\mathbb{N}^*} 3 >_{\mathbb{N}^*} \mathcal{E}\ |\pi_1| >_{\mathbb{N}} |\pi_2|$

Unser Ziel ist also, TESe $\mathcal{R}$ zu verwenden, bei denen $\rightarrow_{\mathcal{R}}$ fundiert ist $t \rightarrow_{\mathcal{R}} \ldots \rightarrow_{\mathcal{R}} q$ und q ist nicht reduzierbar.

Definition 3.3.10: Normalform
Sei $\rightarrow$ eine Relation über Menge M. Ein Objekt $q \in M$ heisst Normalform gdw. es kein Objekt $q' \in M$ gibt mit $q \rightarrow q'$.
Ein Objekt heisst Normalform eines Objektes t gdw $t \rightarrow^* q$ und q Normalform ist. Falls die Normalform eindeutig ist, so bezeichnet $t\downarrow$ die Normalform von t.
Die Relation $\rightarrow$ ist normalisierend, wenn es zu jedem Objekt t mindestens eine Normalform g gibt.
Die Relation $\rightarrow$ ist eindeutig normalisierend, wenn es zu jedem Objekt t genau eine Normalform g gibt.

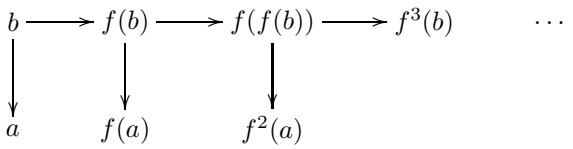


Lemma 3.3.11:
Jede fundierte Relation ist normalisierend.
Beweis: Annahme: es existiert $t \in M$ ohne Normalform.
$t \rightarrow t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow \ldots$
$\lightning$ zur Fundiertheit
□

Definition 3.3.12: Terminierung von TESen
Ein TES $\mathcal{R}$ terminiert gdw. $\rightarrow_{\mathcal{R}}$ fundiert ist.
Ein TES $\mathcal{R}$ ist normalisierend gdw. $\rightarrow_{\mathcal{R}}$ normalisierend ist. (d.h. jeder Term hat mindestens eine Normalform)
Analog „eindeutig Normalisierend“.

Beispiel 3.3.13:

a) $\mathcal{R} = \{b \rightarrow a, b \rightarrow f(b)\}$



[28.05.04]

nicht terminierend, normalisierend, nicht eindeutig normalisierend

b) $\mathcal{R} = \{b \to a, b \to b\}$
nicht terminierend. $(b \to b \to b \to b ...)$ eindeutig normalisierend

c) $\mathcal{R} = \{b \to c, b \to a\}$
terminiert. nicht endlich normalisierend.

d) $\mathcal{R} = \{c \to b, a \to b\}$
terminiert, eindeutig normalisierend

- fundiert/terminiert
- normalisierend (folgt aus Fundiertheit/Terminierung oder eindeutiger Normalisierung)
- eindeutig normalisierend

Einige Einschänkungen an die Gestalt von Regeln sind für Terminierung nötig:

- $\mathcal{V}(r) \subseteq \mathcal{V}(l)$ für alle Regeln $l \to r$:
  Bsp: $a \to f(x) \quad a \to f(a) \to f(f(a)) \to f^3(a) \to \ldots$
  Allgemein: Falls $x \in \mathcal{V}(x) \backslash \mathcal{V}(l) : \sigma = \{x/l\} \quad r|_\pi = x$
  $l \to r\sigma = r\sigma[l]_\pi \to r\sigma[r\sigma]_\pi = r\sigma[r\sigma[l]_\pi]_\pi \to \ldots$
- $l \notin \mathcal{V}$ für alle Regeln $l \to r$:
  Bsp: $x \to f(x)$: $x \to f(x) \to f^2(x) \to f^3(x) \to \ldots$
  Allgemein: Falls es Regel $x \to r$ gibt, gilt: $\sigma_1 = \{x/r\}$, $\sigma_2 = \{x/r\sigma_1\}$, $\sigma_3 = \{x/r\sigma_2\}, \ldots$
  $x \to r \to r\sigma_1 \to r\sigma_2 \to r\sigma_3 \to ...$

Terminierung alleine reicht nicht aus, um das gewünschte Verfahren für das Wortproblem verwenden zu können: $\mathcal{R} = \{b \to c, b \to a\}$ terminiert.
Um zu beweisen, dass $a \equiv_\mathcal{E} c$ gilt: a und c sind schon Normalformen $\curvearrowright$ „False“
Aber: $a \equiv_\mathcal{E} c$ gilt, denn $a \leftrightarrow^*_\mathcal{R} c$: $a \leftarrow_\mathcal{R} b \to_\mathcal{R} c$
Wir wollen aber nicht „$s \leftrightarrow^*_\mathcal{R} t$“ untersuchen, sondern wir untersuchen nur, ob s und t zusammenführbar sind, d.h., ob es einen Term q gibt mit: „$s \leftarrow^*_\mathcal{R} q \to^*_\mathcal{R} t$“
Im allgemeinen ist das eine deutliche Einschränkung $\curvearrowright$ Betrachte nur noch solche TESe, wo „$s \leftrightarrow^*_\mathcal{R} t$“ und Zusammführbarkeit dasselbe sind.

Definition 3.3.14: Zusammenführbarkeit, Church-Rosser-Eigenschaft
Sei $\to$ eine Relation auf Menge M. Zwei Elemente $s, t \in M$ sind zusammenführbar (joinable, schreibweise $s \downarrow t$) gdw. es ein $q \in M$ gibt mit $s \to^* q \leftarrow^* t$.
Die Relation hat die Church-Rosser-Eigenschaft gdw. für alle $s, t \in M$ gilt: wenn $s \leftrightarrow^* t$, dann $s \downarrow t$.

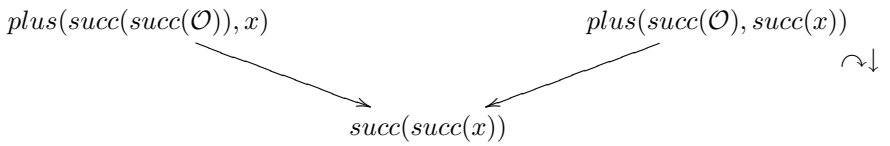
Ein TES $\mathcal{R}$ hat die Church-Rosser-Eigenschaft gdw. $\rightarrow_{\mathcal{R}}$ die Church-Rosser-Eigenschaft hat.

(Church und Rosser haben die Church-Rosser-Eigenschaft des Lambda-Kalküls 1936 gezeigt)

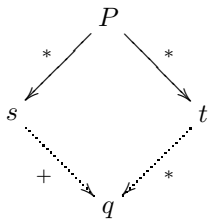
Beispiel 3.3.15:

- $\{b \rightarrow c, b \rightarrow a\}$ hat die Church-Rosser-Eigenschaft nicht: $a \leftrightarrow^*_{\mathcal{R}} c$, aber $a \not\downarrow_{\mathcal{R}} c$
- plus-TES hat die Church-Rosser-Eigenschaft:

  $plus(succ(succ(\mathcal{O})), x) \leftrightarrow^*_{\mathcal{R}} plus(succ(\mathcal{O})succ(x))$ und



Um die Church-Rosser-Eigenschaft zu überprüfen, müsste man unendlich viele Termpaare s, t untersuchen.
Besser für die (automatische) Überprüfung geeignet: Konfluenz.
Konfluenz: Wenn ein Indeterminismus auftritt:

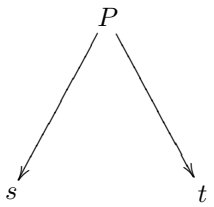


→ Wenn diese Pfeile existieren, dann auch die ⇢

dann ist es egal, für welche welche Alternative man sich entscheidet.

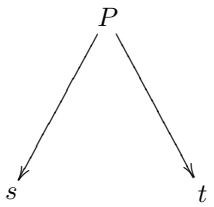
Definition 3.3.16: Konfluenz
Eine Relation $\rightarrow$ über der Menge M heisst Konfluent gdw. für alle $p, s, t \in M$ gilt: Wenn $p \rightarrow^* s$ und $p \rightarrow^* t$, $\underbrace{\text{dann existiert ein } q \in M \text{ mit } s \rightarrow^* q \text{ und } t \rightarrow^* q}_{\text{d.h.} s \downarrow t.}$
Ein TES $\mathcal{R}$ heisst konfluent gdw. $\rightarrow_{\mathcal{R}}$ konfluent ist.

Satz 3.3.17: Church-Rosser gdw. Konfluent
Für alle Relationen $\rightarrow$ gilt: $\rightarrow$ hat die Church-Rosser-Eigenschaft gdw. $\rightarrow$ konfluent ist.
Beweis: „Church-Rosser $\curvearrowright$ Konfluenz“: $\rightarrow$ habe die Church-Rosser-Eigenschaft.
zu Zeigen:



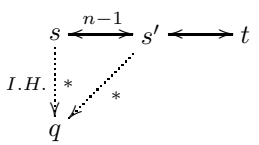
$\curvearrowright s \downarrow t$

Es gilt:

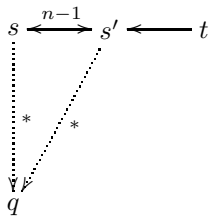


$\curvearrowright s \leftrightarrow^* t \overset{CR}{\curvearrowright} s \downarrow t$

„Konfluenz $\curvearrowright$ Church-Rosser“: Sei $\rightarrow$ Konfluent.
z.Z.: $\underbrace{s \leftrightarrow^* t}_{s \leftrightarrow^n t (n \in \mathbb{N})} \curvearrowright s \downarrow t$. Induktion über n (über die Länge von $s \leftrightarrow \ldots \leftrightarrow t$).
Induktionsanfang: n = 0
$s = t \curvearrowright s \downarrow t \surd$
Induktionsschluss: n > 0:

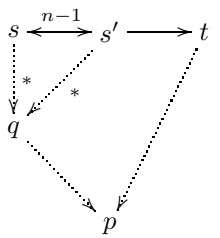


Fall 1



$\curvearrowright s \downarrow t$

Fall 2



$\leftarrow$ wegen der Konfluenz, da q und t einen gemeinsamen Vorgänger s' haben $\curvearrowright s \downarrow t$

Bsp: $\{b \rightarrow c, b \rightarrow a\}$ Problem liegt im Endeffekt daran, dass ein Term (b) 2 Normalformen hat.

Konfluenz garantiert, dass es höchstens eine Normalform gibt. $\curvearrowright$ Konfluenz generell wichtig bei Programmen, da unabhängig von der verwendeten Auswertungsstrategie das Resultat eines Programms eindeutig sein sollte. ($\rightarrow$ vergleiche Diskussionen über „call-by-value“ und „call-by-name“)

Lemma 3.3.18: Konfluenz bedeutet eindeutige Normalformen

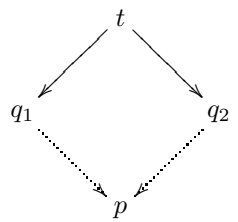
a) Wenn eine Relation $\curvearrowright$ konfluent ist, dann hat jedes Objekt höchstens eine Normalform.

b) Wenn eine Relation $\curvearrowright$ konfluent und normalisierend ist, dann hat jedes Objekt genau eine Normalform. (d.h. $\rightarrow$ ist eindeutig normalisierend)

c) Jede eindeutig normalisierende Relation ist konfluent.

Beweis:

a) Annahme: Es existiert ein Objekt t mit 2 verschiedenen Normalformen q_1 und q_2

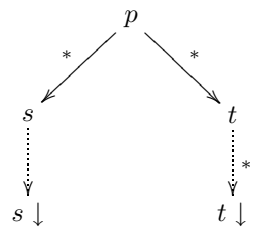


Da q_1 ud q_2 Normalformen sind, gilt $q_1 = p = q_2$ $\lightning$

b) Konfluenz $\curvearrowright$ höchstens eine NF (a) $\surd$

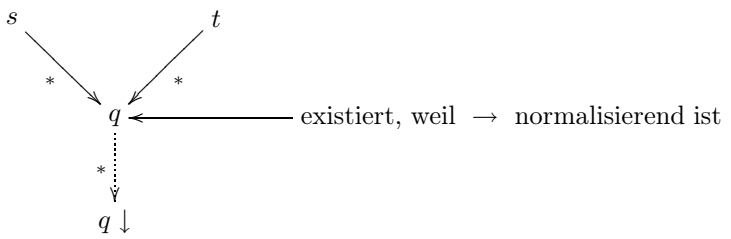
Normalisierend $\curvearrowright$ mindestens eine NF $\surd$

c) Sei $\rightarrow$ eindeutig normalisierend, z.z: $\rightarrow$ ist konfluent.



$s\downarrow$ und $t\downarrow$ existieren, beide sind Normalformen von P. Wegen eindeutig normalisierend: $s\downarrow = t\downarrow$

Satz 3.3.19: Überführbarkeit und Zusammenführbarkeit Sei $\rightarrow$ eine normalisierende konfluente Relation über M, seien $s, t \in M$
Dann gilt: $s \leftrightarrow^* t$ gdw. $s \downarrow = t \downarrow$
Beweis: „$\curvearrowright$“: $s \downarrow = t \downarrow \curvearrowright s \downarrow t \curvearrowright s \leftrightarrow^* t \surd$
„$\curvearrowright$“: $s \leftrightarrow^* t \overset{\text{Satz 3.3.17}^1}{\curvearrowright}$
$s \downarrow t$, d.h. es existiert q mit



$\curvearrowright$ wegen der Eindeutigkeit der Normalformen. $s \downarrow q \downarrow = t \downarrow$ (Lemma 3.3.18)
□

[08.06.04]

Ziel: $s \equiv_{\mathcal{E}} t$, $s, t, \mathcal{E}$ dürfen Variablen enthalten $\curvearrowright$ TES
Eigenschaften:

- Terminierung

  $\downarrow$

- Normalisierung (jeder Term hat mindestens eine Normalform)
- CR-Eigenschaft

  $\updownarrow$ ($\curvearrowright$ jeder Term hat höchstens eine Normalform)

- Konfluenz

Eigentlich müssten wir $s \leftrightarrow^*_{\mathcal{R}} t$ untersuchen.
Wenn $\mathcal{R}$ normalisierend und Konfluent ist, dann reicht es, stattdessen $s \downarrow_{\mathcal{R}} = t \downarrow_{\mathcal{R}}$ zu untersuchen.
Normalisierung alleine hat den Nachteil, dass man $s \downarrow_{\mathcal{R}}$ eventuell nicht berechnen kann, da s auch unendliche Reduktionen haben kann.
Deshalb: Einschränkung auf terminierende und konfluente TESe

Definition 3.3.20:
Ein TES ist konvergent, gdw. es terminiert und konfluent ist.

Beispiel 3.3.21: plusTES ist konvergent
Bei konvergenten TESen kann man das Berechnen von Normalformen als “Interpreter“ des TES-Programms ansehen.

[0] Konfluenz äquivalent zur CR-Eigenschaft

$\curvearrowright$ konvergente TESe können zum „Rechnen“ benutzt werden.
Beispiel 2+1: Rufe „plus“ mit der „rechnerinternen“ Darstellung“ von 2 und 1 auf: plus(s(s($\mathcal{O}$)),s($\mathcal{O}$))
Anwenden des Interpreters “$\downarrow_{\mathcal{R}}$ “ : $plus(s(s(\mathcal{O})), s(\mathcal{O}))\downarrow_{\mathcal{R}} = \underbrace{s(s(s(\mathcal{O})))}_{\text{rechnerinterne Darstellung von 3}}$
Wir wollen konvergente TESe nicht nur zum Rechnen, sondern zum Beweisen benutzen. $\curvearrowright$ Algorithmus WORTPROBLEM

Satz 3.3.22:

a) Der Algorithmus WORTPROBLEM terminiert

b) Falls $\mathcal{R}$ äquivalent zum Gleichungssystem $\mathcal{E}$ ist, dann ist das Algorithmus WORTPROBLEM korrekt

c) Existiert zu einem Gleichungssystem ein äquivalentes konvergentes TES, so ist das Wortproblem über diesem Gleichungsystem entscheidbar

Beweis:

a) folgt aus der Terminierung von $\mathcal{R}$, da jede Reduktionsfolge endlich ist

b) $s \equiv_{\mathcal{E}} \underbrace{\text{gdw}}_{\text{Satz von Birkhoff 3.1.14}} s \leftrightarrow^*_{\mathcal{E}} t \underbrace{\text{gdw.}}_{\text{Äquivalenz von } \mathcal{E} \text{ und } \mathcal{R}} s \leftrightarrow^*_{\mathcal{R}} t \underbrace{\text{gdw.}}_{\text{Satz 3.3.19}\mathcal{R}\text{normalisierend und konfluent}} s\downarrow_{\mathcal{R}} = t\downarrow_{\mathcal{R}}$

c) WORTPROBLEM ist ein Entscheidungsverfahren (alle Schritte des Algorithmus sind effektiv durchführbar)

Um s zu reduzieren, muss man alle (endlich vielen) Teilterme von s und alle (endlich vielen) linken Regelseiten untersuchen und überprüfen, ob die linke Regelseite den Teilterm matcht. Matching kann automatisch durchgeführt werden $\curvearrowright$ Kapitel 5

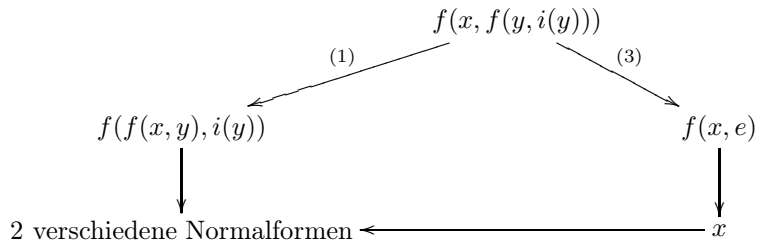
□

Wortproblem ist im Allgemeinen unentscheidbar:

- es existieren Gleichungssysteme $\mathcal{E}$, zu denen kein äquivalentes, konvergentes TES existiert
- selbst wenn es ein äquivalentes, konvergentes TES gibt, so findet man dieses nicht unbedingt $\curvearrowright$ im Folgenden: Wie findet man zu $\mathcal{E}$ ain äquivalentes, konvergentes TES?

Beispiel 3.3.23: Gruppenbeispiel. Wie erhält man ein äquivalentes, konvergentes TES
$f(x, f(y, z)) \rightarrow f(f(x, y), z)$
$f(x, e) \rightarrow x$

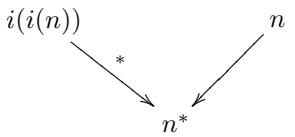
$f(x, i(x)) \to e$
ist äquivalent, terminiert, aber ist nicht konfluent.



Wenn man Wortproblem benutzt, um mit diesem TES $i(i(n)) \equiv_{\mathcal{E}} n$ zu untersuchen, so ergibt sich „ False“ (beide sind Normalformen).
Lösung, um ein konvergentes TES zu erhalten: Starte mit (1), (2), (3) und vervollständige TES um zusätzliche Regeln $l \to r$.

- Äquivalenz muss erhalten bleiben (d.h. es muss $l \equiv_{\mathcal{E}} r$ gelten)
- Terminierung muss erhalten bleiben
- Füge solange neue Regeln hinzu, bis das TES konfluent ist.

Im Beispiel: es existiert ein 10-Regel TES, das zu $\mathcal{E}$ äquivalent und konvergent ist. $\curvearrowright$ Automatischer Beweis, der alle Aussagen über Gruppen entscheiden kann.
Es gilt:



Ziel jetzt: Generiere zu einer Gleichungsmenge automatisch ein äquivalentes, konvergentes TES. (Erzeuge automatisch einen automatischen Beweiser, der Aussagen über der Gleichungsmenge entscheidet)
Vorgehen, um zu $\mathcal{E}$ ein äquivalentes, konvergentes TES zu konstruieren:

1. Konstruiere ein äquivalentes TES, z.B. durch Richten von Gleichungen
2. Überprüfe, ob $\mathcal{R}$ terminiert. Wenn nein, gib auf (oder richte Gleichungen anders)
3. Überprüfe, ob $\mathcal{R}$ konfluent ist. Falls ja: konvergentes äquivalentes TES. Ende mit Erfolg.
4. Falls nein, vervollständige $\mathcal{R}$ um zusätzliche Regeln $l \to r$ wobei $l \equiv_{\mathcal{E}} r$. Zuück zu Schritt 2.

3 Möglichkeiten:

- nach endlich vielen Schritten erhält man äquivalentes konvergentes TES
- Scheitern aufgrund fehlgeschlagenen Terminierungsbeweis ($\rightarrow$ scheitern wird bemerkt, wiederholter Versuch mit anderer Technik für Terminierungsbeweise möglich)
- Verfahren terminiert nicht (Vervollständigung mit unendlich vielen Regeln, scheitern wird nicht bemerkt)

Folgende Fragen sind zu lösen:

- Wie kann man zu einem TES herausfinden, ob es terminiert? (Kapitel 4, unentscheidbar)
- Wie kann man zu einem TES herausfinden, ob es konfluent ist? (Kapitel 5, entscheidbar bei terminierenden TESen)
- Wie kann man ein TES vervollständigen? (Kapitel 6)

Kapitel 4

Terminierung von Termersetzungssystemen

Warum interessant?

1. Nötig, um Wortproblem zu lösen (d.h., um zu einer Gleichungsmenge ein konvergentes, äquivalentes TES zu konstruieren)
2. Nötig, um Konfluenz zu entscheiden (ohne Terminierung ist konfluenz unentscheidbar)
3. Generelle essentielle Eigenschaft korrekter Programme (Terminierungsanalyse grundlegend in Software-Verifikation)
4. Nachweis anderer Programm-Eigenschaften benötigt Induktion. Hierfür braucht man Terminierung des Programmes.

Vorgehen:
4.1 Zusammenhang Terminierung/ Induktion
4.2 Entscheidbarkeitsresultate zur Terminierung (und Verfahren zum automatischen Terminierungsnachweis bei TESen ohne Variablen auf rechten Seiten)
4.3 Ansatz für automatische Terminierungsbeweise
4.4 Automatische Erzeugung von Reduktionsrelationen

4.1 Noethersche Induktion

Bisher: Induktionsbeweise anhand von Datenstrukturen.
Schreibweise: $\forall, \Rightarrow$ auf der Meta-Ebene
Natürliche Zahlen (Peano-Induktion): Induktionsanfang $\varphi(0)$
Induktionsschluss: $\forall y \in \mathbb{N} \quad \varphi(y) \Rightarrow \varphi(y+1)$
Jetzt: ein allgemeineres Induktionsprinzip, so dass alle bisherigen Induktionsprinzipien Spezialfälle davon sind. [11.06.04]
Induktion:

- über beliebige Mengen M ($\mathbb{N}$, $\mathcal{T}(\Sigma, \mathcal{V})$, $\mathbb{N}^*$, ...)
- beliebige Induktionsrelationen $\succ$ (Idee der Induktion: wenn man im Induktionsschritt die Aussage für $m \in \mathbb{N}$ zeigen will, dann darf man sie für alle bezüglich $\succ$ kleineren Objekte vorraussetzen)
- Peano-Induktion: $m \succ k$ gdw. $m = k + 1$ $(m, k \in \mathbb{N})$
- Strukturelle Induktion über Terme: $s \succ t$ gdw. t ist direkter Teilterm von s $(s, t, \in \mathcal{T}(\Sigma, \mathcal{V}))$

Induktionsrelation darf nicht ganz beliebig sein:
Bsp: Induktion auf $\mathbb{N}$, $m \succ k$ gdw. m=k
Zeige: $\forall x \in \mathbb{N}$: $x = 0$
Induktionsschritt: $\forall m \in \mathbb{N}$, $m = 0$, Induktionshypothese: $k = 0$, für alle k mit $m \succ k$
d.h. $\forall m \quad m = 0 \Rightarrow m = 0 \surd$
Bedingung: $\succ$ muss fundiert sein! (d.h. keine ∞ Folgen $m_0 \succ m_1 \succ m_2 \succ \ldots$)

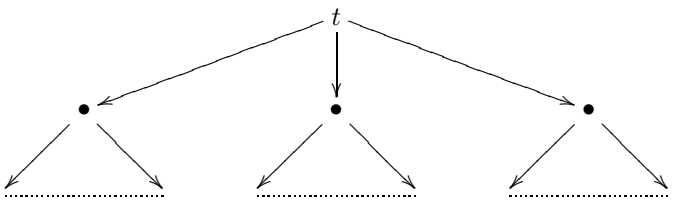
Definition 4.1.1: Noethersches Induktionsprinzip
Sei $\succ$ eine fundierte Relation über einer Menge M. Sei $\varphi(m)$ eine Aussage über Objekte $m \in M$. Falls für alle $m \in M$ gilt: Wenn $\varphi(k)$ für alle $k \in M$ mit $m \succ k$ gilt, dann gilt auch $\varphi(m)$.
Dann gilt $\varphi(m)$ für alle $n \in M$.
D.h.: $(\forall m \in M : \underbrace{(\underbrace{\forall k \in M . m \succ k \Rightarrow \varphi(k)}_{\text{Induktionshypothese}}) \Rightarrow \underbrace{\varphi(m)}_{Induktionskonklusion}}_{(*)}) \Rightarrow \forall n \in M.$
$\varphi(n)$ ("$\Rightarrow$" gilt auch)
In der noetherschen Induktion gibt es keinen Induktionsanfang und Induktionsschluss. Beweisstechnisch ist es oft vorteilhaft, beim Beweis der noetherschen Induktionsformel (*), die Fälle mit $m \in M$, zu dene es kein k mit $m \succ k$ gibt, getrennt zu betrachten. (Diese Fälle entsprechen dem Induktionsanfang)

Satz 4.1.2: Das noethersche Induktionsprinzip ist korrekt
Beweis: Annahme: Noethersche Induktionsformel (*) gilt, aber es existiert ein Gegenbeispiel $n_0 \in M$, so dass $\varphi(n_0)$ nicht gilt.
Wegen (*): $\varphi(n_0)$ wäre wahr, wenn $\varphi(m)$ für alle $n_0 \succ k$ gelten würde. $\curvearrowright$ es existiert also ein kleineres Gegebeispiel als n_0, d.h. ein n_1 mit $n_0 \succ n_1$ und $\varphi(n_1)$ gilt nicht.
Wegen (*): $\varphi(n_1)$ wäre wahr, wenn $\varphi(m)$ für alle $n_1 \succ k$ gelten würde. $\curvearrowright$ es existiert also ein kleineres Gegenbeispiel als n_1, d.h. ein n_2 mit $n_0 \succ n_1 \succ n_2$ und $\varphi(n_2)$ gilt nicht. ... $\Rightarrow$ es existiert eine unendliche absteigende Folge $n_0 \succ n_1 \succ n_2 \succ \ldots$ von Gegenbeispielen ($\varphi(n_i)$ gilt nicht für alle i)
$\lightning$ zur Fundiertheit.
$\square$

Satz 4.1.3: Lemma von König
Anwendungsbeispiel für noethersche Induktion, wird später benötigt
Ein Baum mit endlichem Verzweigungsgrad (d.h. jeder Knoten hat nur endlich viele Kinder), in dem jeder Pfad endlich ist, besitzt nur endlich viele Knoten.
Beweis: Sei B ein Baum wie oben.
Sei M die Menge aller Knoten des Baums. Zu $m \in M$ sei B_m der Teilbaum, der vom Knoten m aufgespannt wird. Sein $\succ$ eine Relation auf Knoten, wobei $m \succ k$ gdw. k (direktes) Kind von m ist. $\succ$ ist fundiert, weil jeder Pfad endlich ist.
Wir zeigen die folgende Aussage $\varphi(n)$ für alle Knoten $n \in M$:
$\varphi(n)$: Der von n aufgespannte Teilbaum B_n hat nur endlich viele Knoten.
Da $\varphi(n)$ auch für die Wurzel n gilt, folgt daraus der Satz.
Beweis mit noetherscher Induktion: Für alle $m \in M$: Wenn $\varphi(k)$ für alle $m \succ k$ gilt, dann gilt $\varphi(m)$. D.h.: Für alle Knoten m: Wenn $\underbrace{|B_k|}_{\text{Anzahl Knoten in} B_k}$ endlich ist für alle direkten Kinder k von M, dann ist auch $\underbrace{|B_m|}_{|B_m|=1+\sum_{\text{k ist direktes Kind von m}}|B_k|}$ endlich $\surd$
B_k ist endlich, weil Verzweigungsgrad endlich ist und weil (wegen Induktionshypothese) $|B_k|$ endlich ist.

4.2 Entscheidbarkeitsresultate zur Terminierung

Satz 4.2.1: Unentscheidbarkeit des Halteproblems für TESe
Sei $\mathcal{R}$ ein TES über Σ und $\mathcal{V}$, sei $t \in \mathcal{T}(\Sigma, \mathcal{V})$. Das Problem, ob t keine unendliche Reduktionen mit $\rightarrow_{\mathcal{R}}$ hat, ist unentscheidbar, aber semi-entscheidbar (Halteproblem). Das Problem, ob $\mathcal{R}$ terminiert, d.h. ob alle Terme nur endlich viele Reduktionen haben, ist nicht einmal semi-entscheidbar (universelles Halteproblem).
Beweisskizze: Für jede Turingmaschine TM kann man ein TES $\mathcal{R}(TM)$ angeben, das TM simuliert. ($\curvearrowright$ TESe sind eine turing-vollständige Programmiersprache). Die Unentscheidbarkeitsaussagen folgen daher aus den Unentscheidbarkeitsaussagen für Turigmaschinen.
Semi-entscheidbarkeit des Halteproblems: Konstruiere Suchbaum



wobei u als Kinder die Knoten $v_1, \ldots, v_n$, wobei $u \rightarrow_{\mathcal{R}} v_i$.

Endlicher Verzweigungsgrad, weil $\mathcal{R}$ endlich und $\mathcal{V}(r) \subseteq \mathcal{V}(l)$ für alle Regeln $l \to r$.
t terminiert gdw. alle Pfade des Suchbaums endlich sind.
Lemma von König $\curvearrowright$ wenn t terminiert, dann hat der Baum endlich viele Knoten $\curvearrowright$ dann terminiert das Semi-entscheidungsverfahren mit Erfolg.
□

Spezialfall, in dem Terminierung von TES entscheidbar ist: TESe ohne Variablen auf rechten Seiten von Regeln.

Beispiel 4.2.2:
and(true, true) → true
and(x, false) → false
and(false, x) → and(true, not(false))
not(false) → true
not(true) → and(false, false)
and(false, false) $\to_{\mathcal{R}}$
and(true, not(true)) $\to_{\mathcal{R}}$
and(true, and(false, false)) $\to_{\mathcal{R}}$
and(true, and(true, not(true))) $\to_{\mathcal{R}}$
and(true, and(true, and(false, false)))

Lemma 4.2.3:
Sei $\mathcal{R}$ ein TES, wobei $\mathcal{V}(r) = \emptyset$ für alle $l \to r \in \mathcal{R}$. Dann terminiert $\mathcal{R}$ gdw. es keine Regel $l \to r \in R$ gibt, so dass $r \to_{\mathcal{R}}^{+} t$ für einen Term t gilt, der r als Teilterm enthält.
Beweis: "⇒": Falls $r \to_{\mathcal{R}}^{+} t$ mit $t \trianglerighteq r$ ($\triangleright$ = echte Teiltermrelation, $\trianglerighteq$ Teiltermrelation) dann terminert $\mathcal{R}$ nicht:
Sei $t|_\pi = r$: $r \to_{\mathcal{R}}^{+} t = t[r$
[15.06.04] $_\pi \to_{\mathcal{R}}^{+} t[t]_\pi = t[t[r]_\pi]_\pi \to_{\mathcal{R}} t[t[r]_\pi]_\pi$ etc.
Spezialfall: TESe ohne Variablen auf rechten Seiten:
$a(t, t) \to t$
$a(x, f) \to f$
$a(f, x) \to a(t, n(t))$
$n(f) \to t$
$n(t) \to f$
$a(t, n(t)) \to a(t, a(f, f)) \to a(t, a(t, n(t)))$
Beweis von Lemma 4.2.3 "⇐":
Induktion über die Anzahl der Regeln in $\mathcal{R}$:
Induktionsanfang: $\mathcal{R} = \emptyset$ trivial, $\mathcal{R}$ terminiert
Induktionsschluss: $\mathcal{R} \neq \emptyset$ Annahme: r terminiert nicht. Dann existiert ein minimaler Term s, der eine unendliche Reduktion hat (d.h. alle echten Teilterme von s terminieren) $s = s_0 \to_{\mathcal{R}} s_1 \to_{\mathcal{R}} s_2 \to_{\mathcal{R}} \ldots$ Aufgrund der Minimalität muss in der Auswertung irgendwann eine Regel an der obersten Position $\mathcal{E}$ angewendet werden. Es existiert ein s_i mit $s_i = l\sigma, s_{i+1} = r\sigma$ für eine Regel

$l \to r \in \mathcal{R}$. Da $\mathcal{V}(r) = \emptyset$, ist $s_{i+1} = r$. Es existiert eine unendliche Reduktion $r = s_{i+1} \to s_{i+2} \to \ldots$, die mit der rechten Seite einer Regel startet.

Fall 1 Die Regel $l \to r$ wird in der unendliche Reduktion $r \to s_{i+2} \to s_{i+3} \to \ldots$ nicht mehr verwendet.

I.H.: $\mathcal{R}\backslash\{l \to r\}$ terminiert. $\lightning$

Fall 2 Die Regel $l \to r$ wird in $r \to s_{i+2} \to s_{i+3} \to \ldots$ verwendet.

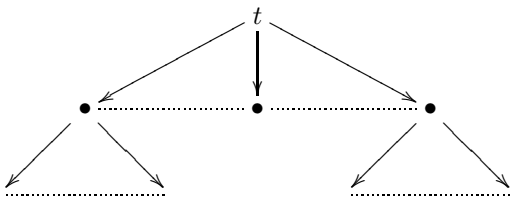
Es existiert also $j \geq i+1$ mit $r \to^* s_j = s_j[l\sigma']_\pi \to s_j[\underbrace{r\sigma'}_{r}]_\pi = s_{j+1} \to s_{j+2} \to \ldots$

Also: $r \to^+_{\mathcal{R}} s_j[r]_\pi$

$\lightning$ zur Vorraussetzung.
□

Idee für Entscheidungsverfahren für Terminierung:

- Konstruiere Suchbäume für alle rechten Seiten



(u hat Kind v gdw. $u \to_{\mathcal{R}} v$)

- (a) Wenn Suchbaum für r einen Term $\underbrace{t}_{\text{nicht an Wurzel}}$ mit $t \trianglerighteq r$ enthält, dann bricht man ab, $\mathcal{R}$ terminiert also nicht.
- (b) Wenn alle Suchbäume endlich sind, dann stoppt man, $\mathcal{R}$ terminiert

(Einer der Fälle a, b tritt immer auf.)
Unterschied zur Terminierung bei allgemeinen TESen:

- bei allgemeinen TESen folgt aus der Terminierung der rechten Seiten nicht die Terminierung des TES. ($\mathcal{R} = \{f(a,x) \to f(x,x)\}$ $f(a,a) \to f(a,a) \to \ldots$ terminiert nicht, Aber die rechte Seite f(x,x) ist Normalform)
- selbst wenn die rechte Seite r eine unendliche Auswertung hat, muss es keinen Term t mit $r \to^+ t$ und $t \trianglerighteq r$ geben. ($\mathcal{R} = \{f(x) \to f(s(x))\}$ $\underbrace{f(s(x))}_{\text{tritt in der Reduktion nie wieder als Teilterm auf}} \to f(s(s(x))) \to f(s^3(x)) \to \ldots$)

[Folie Right-ground-Terminierung]
T_i = Blätter des i-ten Suchbaums (für jede rechte Seite S existiert ein Suchbaum)
Schritt 2: Konstruiere die nächste Ebene die Suchbäume
$T_1 = \{t\}$ $T_2 = \{f\}$ $T_3 = \{a(t, n(t))\}$ $T_4 = \{t\}$ $T_5 = \{a(f, f)\}$
$T_1 = \emptyset$ $T_2 = \emptyset$ $T_3 = \{a(t, a(f, f))\}$ $T_4 = \emptyset$ $T_5 = \{f, a(t, n(t))\}$
$T_1 = \emptyset$ $T_2 = \emptyset$ $T_3 = \{a(t, a(t, a(t, n(t))))\}$ $T_4 = \emptyset$ $T_5 = \{a(t, a(f, f))\}$
$\Rightarrow$ "False"

Satz 4.2.4:
Sei $\mathcal{R}$ ein TES 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:

Fall 1 $\mathcal{R}$ terminiert.

Dann existiert keine rechte Seite r_i mit $r_i \to^+ t \trianglerighteq r_i$. $\curvearrowright$ Algorithmus gibt nicht "False" aus.

Kein r_i hat unendlich lange Reduktionen und der Suchbaum, der vom Algorithmus konstruiert wird, hat endlichen Verzweigungsgrad. $\overset{\text{Lemma von König}}{\curvearrowright}$ Suchbaum hat nur endlich viele Knoten $\curvearrowright$ Algorithmus terminiert (und gibt "true" aus).

Fall 2 $\mathcal{R}$ terminiert nicht

Wegen Lemma 4.2.3 existiert ein $r_i \to^+ t \trianglerighteq r_i$.Nach endlich vielen Schritten gilt $t \in T_i$ und der Algorithmus terminiert mit "False".

□

4.3 Terminierung mit Reduktionsrelationen

Jetzt: untersuche Terminierung von beliebigen TESen (mit Variablen rechts)
$\mathcal{R}$ terminiert gdw. $\to_{\mathcal{R}}$ fundiert ist.
Grundidee: Nimm eine Relation $\succ$ auf Termen, von der man weiß, dass sie fundiert ist, und zeige, dass aus $s \to_{\mathcal{R}} t$ jeweils $s \succ t$ folgt. Da man dies automatisch überprüfen will, kann man nicht alle unendlich vielen Terme s und t untersuchen.
Lösung: Zeige nur $l \succ r$ für alle (endlich vielen) Regeln $l \to r$. Reicht das, um sicher zu stellen, dass dann aus $s \to_{\mathcal{R}} t$ auch $s \succ t$ für alle Terme s, t folgt (d.h. reicht das für die Terminierung?)?
Bsp: $endless(x) \to endless(s(x))$
Definieren $\succ$ wie folgt: $t_1 \succ t_2$ gdw. $t_1 = endless(x)$ und $t_2 = endless(s(x))$
Es gilt also: $endless(x) \succ endless(s(x)) \not\succ endless(s(s(x)))$
Problem: $\succ$ ist nicht stabil! (Aber da Regeln in beliebigen Instantiierungen angewendet werden dürfen, sollte aus $l \succ r$ auch $l\sigma \succ r\sigma$ für alle σ folgen).

Lösung: Finde fundierte, stabile Relation $\succ$ und zeige $l \succ r$ für alle Regeln $l \to r$.
Bsp: Echte Teiltermrelation $\rhd$ ($s \rhd t \curvearrowright s\sigma \rhd t\sigma$)
Reicht das für die Terminierung?
Bsp: $infty(x) \to s(infty(x))$
Definieren $\succ$ wie folgt: $t_1 \succ t_2$ gdw. $t_1 = infty(\ldots)$ und $t_2 = s(\ldots)$

- $\succ$ ist fundiert ($infty(\ldots) \succ s(\ldots)$)
- $\succ$ ist stabil

Problem: $\succ$ ist nicht monoton! (Aus $l \succ r$ sollte $q[l\sigma]_\pi \succ q[r\sigma]_\pi$ folgen: $\succ$ muss stabil und monoton sein).
$\rhd$: fundiert, stabil, nicht monoton. ($f(x) \rhd x, s(f(x)) \not\rhd s(x)$)
$\to_{||}$: $t_1 \to_{||} t_2$, falls t_1 mehr Symbole enthält als t_2: fundiert, nicht stabil($f(x) \to_{||} y, f(x)\sigma \not\to_{||} y\sigma$ wenn $\sigma(x) = x, \sigma(y) = s(s(s(s(y)))))$, monoton
Idee für fundierte, stabile, monotone Relation: Modifiziere $\rhd$ so, dass sie auch monoton wird $\Rightarrow$ Einbettungsordnung

Definition 4.3.1:
Für eine Signatur Σ ist die Einbettungsordnung $\succ_{emb}$ ("embedding order") über $\mathcal{T}(\Sigma, \mathcal{V})$ definiert als $s \succ_{emb} t$ gdw.

- s. Folie (t ist echter Teilterm von s oder kleiner als ein echter Teilterm von s)
- s. Folie (Monotonie)

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

$s(plus(infty(s(x)), y)) \succ_{emb} plus(infty)(x), y)$, denn
$plus(infty(s(x)), y) \succ_{emb} plus(infty(x), y)$ (Punkt 1), denn
$infty(s(x) \succ_{emb} infty(x)$ (Punkt 2), denn
$s(x) \succ_{emb} x$ (Punkt 2), denn
$x \succeq_{emb} x$ (Punkt 1) [18.06.04]
Terminierung beliebiger TESe: Zeige $l \succ r$ für alle Regeln $l \to r \in \mathcal{R}$
(Ziel: $s \to_{\mathcal{R}} t \curvearrowright s \succ t$)
$\succ$: fundiert, stabil, monoton

$$\left.\begin{array}{r} \rhd\text{: fundiert, stabil, nicht monoton} \\ \to_{||}\text{ : fundiert, nicht stabil, monoton} \end{array}\right\} \curvearrowright \text{Einbettungsordnung}$$

Terminierung lässt sich jetzt mit Hilfe der Einbettungsordnung automatisch nachweisen (denn $s \succ_{emb} t$ lässt sich automatisch untersuchen – Definition von $\succ_{emb}$ lässt sich direkt in einen rekusiven Algorithmus übersetzen)
$m(x, \mathcal{O}) \succ_{emb} x$, denn $x \succeq_{emb} x$ (wegen Punkt 1)
$m(\mathcal{O}, s(y)) \succ_{emb} \mathcal{O}$, denn $\mathcal{O} \succeq_{emb} \mathcal{O}$ (wegen Punkt 1)
$m(s(x), s(y)) \succ_{emb} m(x, y)$, denn $s(x) \succ_{emb} x$, $s(y) \succ_{emb} y$ (wegen Punkt 2)

Lemma 4.3.2: Die Einbettunsgordnung ist fundiert, stabil, monoton und transitiv
Beweis:
Fundiertheit: $\succ_{emb} \subseteq \rightarrow_{||}$ (d.h. wenn $s \succ_{emb} t$, dann enthält s mehr Symbole als t). Lässt sich leicht durch strukturelle Induktion zeigen. Da $\rightarrow_{||}$ fundiert ist, ist auch $\succ_{emb}$ fundiert.
Stabilität: z.Z.: $s \succ_{emb} t \curvearrowright s\sigma \succ_{emb} t\sigma$. Strukturelle Induktion über s. Fallunterscheidung anhand der Definition von $\succ_{emb}$

Fall 1 $s = f(s_1, \ldots, s_n)$, $s_i \succeq_{emb} t$

$\overset{I.H.}{\curvearrowright} s_i\sigma \succeq_{emb} t\sigma$

$\curvearrowright s\sigma = f(s_1\sigma, \ldots, s_i\sigma, \ldots, s_n\sigma) \succ_{emb} t\sigma$, denn $s_i\sigma \succeq_{emb} t\sigma$ (Punkt 1)

Fall 2 $s = f(s_1, \ldots, s_n)$, $t = f(t_1, \ldots, t_n)$, $s_i \succ_{emb} t_i$, $s_j \succeq_{emb} t_j$ für alle $j \neq i$

$\overset{I.H.}{\curvearrowright} s_i\sigma \succeq_{emb} t_i\sigma$, $s_j\sigma \succeq_{emb} t_j\sigma$ für alle $j \neq i$

$\overset{Punkt2}{\curvearrowright} s\sigma \succ_{emb} t\sigma$

Beweis für Monotonie und Transitivität analog.

□

Definition 4.3.3:
Eine Relation $\succ$ über Termen, die fundiert, stabil und monoton ist, heisst Reduktionsrelation.
Eine transitive Reduktionsrelation heisst Reduktionsordnung

Ordnung = transitive und antisymmetrische Relation.
(Antisymmetrie: $s \succ t$, $t \succ s \curvearrowright s = t$)
Aus Fundiertheit folgt Irreflexivität und Asymmetrie.
(Asymmetrie: $s \succ t \curvearrowright t \not\succ s$)
Aus Asymmetrie folgt Antisymmetrie.

Satz 4.3.4: Terminierungskriterium mit Reduktionsrelationen, Manna und Ness 1970
Ein TES $\mathcal{R}$ terminiert gdw. es eine Reduktionsrelation $\succ$ gibt, so dass $l \succ r$ für alle Regeln $l \rightarrow r \in \mathcal{R}$ gilt.
Beweis: "$\Leftarrow$" Sei $\succ$ eine Reduktionsrelation, so dass $l \succ r$ für alle Regeln $l \rightarrow r$
Dann folgt für alle Terme s, t aus $s \rightarrow_{\mathcal{R}} t$, dass $s \succ t$ gilt.
(Grund: $s \rightarrow_{\mathcal{R}} t \curvearrowright$ es existiert $\pi \in Occ(s)$ und $\sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})$ mit $s|_\pi = l\sigma$, $t = s[r\sigma]_\pi$ für eine Regel $l \rightarrow r \in \mathcal{R}$.
Es gilt $l \succ r \overset{\text{Stabil}}{\curvearrowright} l\sigma \succ r\sigma \overset{\text{Monoton}}{\curvearrowright} \underbrace{s[l\sigma]_\pi}_{s} \succ \underbrace{s[r\sigma]_\pi}_{t}$.)
Falls $\mathcal{R}$ nicht terminieren würde, gäbe es eine unendliche Reduktion: $t_0 \rightarrow_{\mathcal{R}} t_1 \rightarrow_{\mathcal{R}} t_2 \rightarrow_{\mathcal{R}} \ldots$
$\curvearrowright t_0 \succ t_1 \succ t_2 \succ \ldots$

$\not\downarrow$ zur Fundiertheit von $\succ$
"$\Rightarrow$": $\mathcal{R}$ terminiert: Finde Reduktionsrelation $\succ$ mit $l \succ r$ für alle Regeln $l \to r \in \mathcal{R}$.
Lösung: $\succ := \to_{\mathcal{R}}$
(

- $l \to_{\mathcal{R}} r$
- $\to_{\mathcal{R}}$ fundiert, weil $\mathcal{R}$ terminiert
- $\to_{\mathcal{R}}$ stabil und monoton (Lemma 3.1.13: jede Ersetzungsrelation ist monoton und stabil)

) $\square$

Automatische Termverfahren:

- Wähle eine Reduktionsrelation $\succ$, bei der entscheidbar ist, ob $s \succ t$ gilt (für alle s, t)
- Untersuche, ob $l \succ r$ für alle Regeln $l \to r \in \mathcal{R}$ gilt.
- Wenn ja: "TES $\mathcal{R}$ terminiert"

(Falls $l \not\succ r$ für eine Regel, so folgt daraus nicht, dass $\mathcal{R}$ nicht terminiert. Es könnte eine andere Reduktionsrelation $\succ'$ geben, bei der $l \succ' r$ für alle Regeln gilt.)
Bsp. plus: $plus(\mathcal{O}, y) \succ_{emb} y$
$plus(s(x), y) \not\succ_{emb} s(plus(x, y)) \Rightarrow$ Terminierungsnachweis mit $\succ_{emb}$ scheitert $\curvearrowright$ Verfahren ist zu schwach

4.4 Simplifikationsordnungen und rekursive Pfadordnungen

Einbettungsordnung: 1. Schritt zur Entwicklung von Reduktionsordnungen, die zum Terminierungsnachweis geeignet sind. Entwickle stärkere Ordnungen, die die Einbettungsordnung enthalten.

Definition 4.4.1:
Eine Reduktionsordnung $\succ$ mit $\succ_{emb} \subseteq \succ$ (d.h.: $s \succ_{emb} t \curvearrowright s \succ t$) heisst Simplifikationsordnung.

Wenn man eine neue Relation $\succ$ definiert, muss man anschliessend zeigen, dass es wirklich eine Reduktionsordnung ist. Wenn man weiss, das $\succ$ die Einbettungsordnung enthält, reicht es, statt der Fundiertheit nur die Irreflexivität zeigen.

Satz 4.4.2: Satz von Kruskal, 1960

a) Für jede unendliche Folge von Grundtermen $t_0, t_1, \ldots$ existieren t_i, t_j mit $i < j$, so dass $t_i \preceq_{emb} t_j$

b) Jede stabile, monotone, transitive Relation $\succ$, die die Teiltermeigenschaft $f(x_1, \ldots, x_n) \succ x_i$ für alle $f \in \Sigma$ und alle $1 \leq i \leq n$ erfüllt, enthält die Einbettungsordnung (d.h. $\succ_{emb} \subseteq \succ$)

c) Jede stabile, monotone, transitive, irreflexive Relation $\succ$, die die obige Teiltermeigenschaft erhält, ist fundiert (und damit eine Simplifikationsordnung).

(d.h. um zu zeigen, dass $\succ$ eine Simplifikationsordnung ist, muss man nur diese 5 Fakten nachweisen).
Beweis:

a) ist der eigentliche Satz von Kruskal ($\rightarrow$ Literatur)

b) zu Zeigen: $s \succ_{emb} t \curvearrowright s \succ t$ (sehr leicht, strukturelle Induktion über s)

c) Annahme: $\succ$ nicht fundiert (aber stabil, monoton, transitiv, irreflexiv, hat die Teiltermeigenschaft)

   $\curvearrowright$ es existiert $t_0 \succ t_1 \succ t_2 \succ \ldots$

   Sei σ eine Substitution, die alle Variablen der t_i durch Grundterme instantiiert. (Man kann leicht zeigen, dass $V(t_0) \supseteq V(t_1) \supseteq \ldots \Rightarrow \mathrm{DOM}(\sigma)$ ist endlich)

   $\overset{\text{stabil}}{\curvearrowright}\ t_0\sigma \succ t_1\sigma \succ t_2\sigma \succ \ldots$, alle $t_i\sigma$ sind Grundterme.

   $\overset{(a)}{\curvearrowright}$ es existiert $i < j$ mit $t_i\sigma \succeq_{emb} t_j\sigma \overset{\succ_{emb} \subseteq \succ \text{wegen (b))}}{\curvearrowright}\ t_i\sigma \succeq t_j\sigma$ (1)

   Andererseits: $t_i\sigma \succ \ldots \succ t_j\sigma$

   Transitivität: $t_i\sigma \succ t_j\sigma$ (2)

   $t_i\sigma \overset{(i)}{\succ} t_1\sigma \succeq t_i\sigma \overset{\text{Transitivität}}{\curvearrowright}\ t_1\sigma \succ t_i\sigma$

   $\lightning$ zur Irreflexivität

□

[22.06.04]

$\mathcal{R}$ TES terminiert gdw. es eine Reduktionsrelation $\succ$ gibt, mit $l \succ r$ für alle $l \rightarrow r \in \mathcal{R}$
$\succ_{emb}$ ist Reduktionsordnung, aber zu schwach.
$\Rightarrow$ Definition stärkerer Reduktionsordnungen $\succ$ (mit $\succ_{emb} \subseteq \succ$): Simplifikationsordnungen
1. solche Ordnung: lexikographische Pfadordnung (lpo)
Zum Vergleich von $f(s_1, \ldots s_n)$ und $f(t_1, \ldots, t_n)$ muss man die Argumenten tupel $(s_1, \ldots s_n)$ und $(t_1, \ldots, t_n)$ vergleichen.
Also: Wie kann man eine Relation auf Termen zu einer Relation auf Tupeln erweitern?
Hier: 2 Möglichkeiten: vergleiche Tupel $\underbrace{\text{lexikographisch}}_{\text{(lpo)}}$ oder als $\underbrace{\text{Multimengen}}_{\text{rekursive Pfadordnung (rpo)}}$

Definition 4.4.3: Lexikographische Konbination von Relationen

Sei $\succ_1$ eine Relation auf einer Menge T_1 ($\succ \subseteq T_1 \times T_1$) unde $\succ_2$ eine Relation auf T_2. Dann wird die lexikographische Kombination $\succ_{1\times 2}$ auf $T_1 \times T_2$ wie folgt definiert:
$(s_1, s_2) \succ_{1\times 2} (t_1, t_2)$ gdw. $s_1 \succ_1 t_1$ oder ($s_1 = t_1$ und $s_1 \succ_2 t_2$)
Analog dazu kann man die lexikographische Kombination von n Relationen $\succ_1$,$\succ_1, \ldots, \succ_n$ definieren:
$(s_1, \ldots, s_n) \succ_{1\times\ldots\times n} (t_1, \ldots, t_n)$ gdw. es existiert $i \in \{1, \ldots, n\}$ mit $s_i \succ_i t_i$ und $s_j = t_j$ für alle $1 \leq j < i$.
Die n-fache lexikographische Kombination einer Relation $\succ$ mit sich selbst wird als $\succ^n_{lex}$ bezeichnet.

Beispiel 4.4.4:
$(3,5)(>_{\mathbb{N}})^2(2,6)$, $(3,5)(>_{\mathbb{N}})^2(3,4)$, $(3,5)(\not>_{\mathbb{N}})^2(4,1)$
Im Lexikon: $>_{alph}$ ($a >_{alph} b >_{alph} c >_{alph} \ldots$)
hans $>^4_{alph}$ hugo $>^4_{alph}$ kurt
Fundiertheit bleibt bei lexikographischen Kombinationen erhalten. Hierbei ist es wichtig, dass die Grösse der Tupel, die verglichen werden, nicht wächst. Die Relation in Lexika hat diese Beschränkung nicht und ist deshalb nicht fundiert.
a > ba > bba > bbba > ...

Satz 4.4.5: Fundiertheit bleibt bei lexikographischen Kombinationen erhalten
Sei $\succ_1$ eine Relation auf der nicht-leeren Menge T_1, sei $\succ_2$ eine Relation auf der nicht-leeren Menge T_2.
Dann gilt: $\succ_1$ und $\succ_2$ sind fundiert gdw. ihre lexikographische Kombination $\succ_{1\times 2}$ fundiert ist.
Beweis: "$\Leftarrow$": $\succ_{1\times 2}$ sei fundiert.
$\succ_1$ ist auch fundiert: Falls nicht, dann existiert $u_1 \succ_1 u_2 \succ_1 u_3 \succ_1 \ldots$
Sei $v \in T_2$ beliebig $(u_1, v) \succ_{1\times 2} (u_2, v) \succ_{1\times 2} \ldots$
$\lightning$ zur Fundiertheit.
$\succ_2$ ist auch fundiert: Falls nicht, dann existiert $v_1 \succ_1 v_2 \succ_1 v_3 \succ_1 \ldots$
Sei $u \in T_1$ beliebig $(u, v_1) \succ_{1\times 2} (u, v_2) \succ_{1\times 2} \ldots$
$\lightning$ zur Fundiertheit.
"$\Rightarrow$": $\succ_1$, $\succ_2$ seien fundiert.
Annahme: es existiert eine unendliche Folge $(u_1, v_1) \succ_{1\times 2} (u_2, v_2) \succ_{1\times 2} (u_3, v_3) \succ_{1\times 2} \ldots$
Wir wissen $u_1 \succeq_1 u_2 \succeq_1 u_3 \succeq_1 \ldots$
Da $\succ_1$ fundiert ist, existiert ein i mit $u_i = u_j$ für alle $j \geq i$.
$(u_i, v_1) \succ_{1\times 2} (u_i, v_{i+1}) \succ_{1\times 2} (u_i, v_{i+2}) \succ_{1\times 2} \ldots$
$\curvearrowright v_i \succ_2 v_{i+1} \succ_2 v_{i+2} \succ_2 \ldots$
$\lightning$ zur Fundiertheit von $\succ_2$
□

Aus Satz folgt: Wenn $\succ$ eine fundierte Relation auf Termen ist, dann ist $\succ^n_{lex}$ eine fundierte Relation auf n-Tupeln von Termen.
$\Rightarrow$ Definieren die lpo: $s \succ_{lpo} t$ – wann soll dies gelten?

1. Es soll gelten, wenn ein direkter Teilterm von s grösser oder gleich t ist (wie bei $\succ_{emb}$)

2. Beim Vergleich von $s = f(s_1, \ldots, s_n)$ und $t = f(t_1, \ldots, t_n)$: Vergleiche die Tupel $(s_1, \ldots, s_n)$ und $(t_1, \ldots, t_n)$ lexikographisch, d.h. $(s_1, \ldots, s_n)(\succ_{lpo}^{n}$ $(t_1, \ldots, t_n)$.

   Problem: So wird die Ordnung nicht fundiert: $f(s(\mathcal{O}), \mathcal{O}) \succ f(\mathcal{O}, f(s(\mathcal{O}), \mathcal{O}))$ (1)

   denn $s(\mathcal{O}) \succ \mathcal{O}$ (denn $\mathcal{O} \succeq \mathcal{O}$). Andererseits: $\underbrace{f(\mathcal{O}, f(s(\mathcal{O}), \mathcal{O})) \succ f(s(\mathcal{O}), \mathcal{O})}_{(2)}$,

   denn $f(s(\mathcal{O}), \mathcal{O}) \succeq f(s(\mathcal{O}), \mathcal{O})$

   (1) und (2) verletzen Fundiertheit

   Abhilfe: Verlange, dass die Teilterme $t_{i+1}, \ldots, t_n$ rechts wenigstens kleiner als die gesamte linke Seite $f(s_1, \ldots, s_n)$ sind.

3. Ein wesentlicher Grund für die Schwäche von $\succ_{emb}$ ist, dass man Terme mit unterschiedliche äussersten Funktionssymbol schlecht vergleichen kann. Wie vergleicht man $s = f(s_1, \ldots, s_n)$, $t = g(t_1, .., t_m)$. Verwende hierzu eine zugrundeliegende Ordnung] auf Funktionssymbolen ("Präzedenz") Überprüfe dann, on $f]g$ gilt.

   Problem: zerstört Fundiertheit: Sei f g: $f(x) \succ g(f(x))$ (1)

   Andererseits: $\underbrace{g(f(x)) \succ f(x)}_{(2)}$, denn $f(x) \succeq f(x)$.

   (1) und (2) zerstören Fundiertheit

   Abhilfe: Verlange zusätzlich, dass die Teilterme $t_1, \ldots, t_m$ rechts wenigstens kleiner als die gesamte linke Seite $f(s_1, \ldots, s_n)$ sind.

Definition 4.4.6: Lexikographische Pfadordnung, Kamin und Lévy. 1980

Für eine Signatur Σ und eine fundierte Ordnung] auf Σ ("Präzedenz") ist die lexikographische Pfadordnung (lpo) über $\mathcal{T}(\Sigma, \mathcal{V})$ definiert als $s \succ_{lpo} t$ gdw. s. [Folie]

Beispiel 4.4.7:

Verwende lpo für Terminierungsbeweise.

$plus(\mathcal{O}, y) \to y$	$l \succ_{lpo} r$, denn $y \succeq_{lpo} y$	
	(folgt auch, da $l \succ_{emb} r$)	
$plus(s(x), y) \to s(plus(x, y))$	Wähle Präzedenz mit $plus]s$.	
	z.Z.: $plus(s(x), y) \succ_{lpo} plus(x, y)$	
$times(\mathcal{O}, y) \to \mathcal{O}$	$l \succ_{lpo} r$, da $l \succ_{emb} r$ denn $s(x) \succ_{lpo} x$	
$times(s(x), y) \to plus(y, times(x, y))$	wähle Präzedenz $times]plus$	
	z.Z. $times(s(x), y) \succ_{lpo} y$	
	$times(s(x), y) \succ_{lpo} y$	$\leftarrow$ gilt bereits bei $\succ_{emb}$
	$times(s(x), y) \succ_{lpo} times(x, y)$	$\leftarrow$ gilt bereits bei $\succ_{emb}$

Für Terminierungsbeweise mit lpo kann man eine beliebige (fundierte) Präzedenz verwenden.
Um Terme mit lpo zu untersuchen, könnte man alle (endlich vielen) Präzedenzen durchprobieren.
Besser: Lasse Präzedenz offen und erweitere sie nur dann, wenn dies zum beweisen von $l \succ r$ nötig ist.
Bei der Erweiterung von] muss jeweils die Fundiertheit von] sichergestellt sein.

Beispiel 4.4.8:

$sum(\mathcal{O}, y) \to y$	$l \succ_{lpo} r$, da $l \succ_{emb} r$
$sum(s(x), y) \to sum(x, (s(y))$	$l \succ_{lpo} r : s(x) \succ_{lpo} x$
	$sum(s(x), y) \succ_{lpo} s(y)$
	wähle Präzedenz $sum]s$
	z.Z. $sum(s(x), y) \succ_{lpo} y$

[25.06.04]

Ziel: Zeige lpo ist eine Simplifikations-Ordnung.
Satz von Kruskal: es reicht, folgendes zu zeigen: stabil, monoton, transitiv, Teiltermeigenschaft ($f(x_1, \ldots, x_n) \succ x_i$), irreflexiv

Satz 4.4.9: Die lpo ist eine Simplifikationsordnung
Beweis: Teiltermeigenschaft: $F(x_1, \ldots, x_i, \ldots, x_n) \succ_{lpo} x_i$, denn $x_i \succeq x_i$ (Punkt 1 von Definition 4.4.6)
Stabilität: z.Z.: Falls $s \succ_{lpo} t$, dann $s\sigma \succ_{lpo} t\sigma$ (*)
Induktion über s, t,. Induktionsrelation: $\triangleright^2_{lex}$ (fundiert, Satz 4.4.5) $\curvearrowright$ noethersche Induktion möglich.
D.h.: Wenn wir (*) für s, t zeigen wollen, dürfen wir sie für die Terme s', t' als Induktionshypothese voraussetzen, bei denen $s \triangleright s'$ oder ($s = s'$ und $t \triangleright t'$) gilt.
Fallunterscheidung anhand der Definition der lpo: $s \succ_{lpo} t$

Fall 1 $s = f(s_1, \ldots, s_i, \ldots, s_n)$, $s_i \succeq_{lpo} t$
Induktionshypothese: $s_i\sigma \succeq_{lpo} t\sigma$
$s\sigma = f(s_1\sigma, \ldots, s_i\sigma, \ldots, s_n\sigma) \succ_{lpo} t\sigma$ (Punkt 1 der Definition 4.4.6)

Fall 2 $s = (f(s_1, \ldots, s_n),\ t = g(t_1, \ldots, t_m),\ f]g,\ s \succ_{lpo} t_j \forall j \in \{1, \ldots, m\}$

Induktionshypothese: $s\sigma \succ_{lpo} t_j\sigma$

$s\sigma = f(s_1\sigma, \ldots, s_n\sigma),\ t\sigma = g(t_1\sigma, \ldots, t_m\sigma),\ f]g,\ s\sigma \succ_{lpo} t_j\sigma$ für alle $j \in \{1, \ldots, m\} \curvearrowright s\sigma \succ_{lpo} t\sigma$ (Punkt 2 von Definition 4.4.6)

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

Induktionshypothese: $s_i\sigma \succ t_i\sigma,\ s\sigma \succ t_j\sigma \curvearrowright s\sigma \succ_{lpo} t\sigma$ (Punkt 3 von Definition 4.4.6)

Transitivität: z.Z. wenn $s \succ_{lpo} t$ und $t_i \succ_{lpo} r$, dann $s \succ_{lpo} r$
Induktion über s, t, r. Induktionsrelation: $\rhd^3_{lex}$ (Übung)
Monotonie: z.Z. $s \succ_{lpo} t \curvearrowright q[s]_\pi \succ_{lpo} q[t]_\pi$
Induktion über π (Übung)
Irreflexivität: z.Z. für alle Terme s gilt: $s \not\succ_{lpo} s$
Strukturelle Induktion über s
Annahme: $s \succ_{lpo} s$. Fallunterscheidung anhand Definition 4.4.6

Fall 1 $s = f(s_1, \ldots, s_i, \ldots, s_n),\ s_i \succeq_{lpo} s$ (da $s_i \neq s$ gilt $s_i \succ_{lpo} s$)

Es gilt aber auch: $s \succ_{lpo} s_i \quad s_i \succ_{lpo} s \succ_{lpo} s_i \overset{\text{Transitivität}}{\curvearrowright} s_i \succ_{lpo} s_i$

$\lightning$ zur Induktionshypothese

Fall 2 nicht anwendbar, da dann s 2 verschiedene äussere Funktionssymbole f, g haben müsste.

Fall 3 $s = f(s_1, \ldots, s_i, \ldots, s_n),\ s_i \succ_{lpo} s_i$

$\lightning$ zur Induktionshypothese.

□

LPO darf für Terminierungsbeweise verwendet werden, mit beliebiger Präzedenz (fundierte Ordnung auf Funktionssymbolen).
Automatisierung: beginne mit unbestimmter Präzedenz und erweitere sie, falls nötig. Hierbei muss Fundiertheit der Präzedenz sicher gestellt sein. → Die Frage, ob ein TES mit einer lpo terminiert, ist entscheidbar, aber NP-vollständig. ⇒ es existieren terminierende TES, deren Terminierung nicht mit lpo gezeigt werden kann.

$pred(\mathcal{O}) \to \mathcal{O}pred(succ(x)) \to xminus(x, \mathcal{O}) \to xminus(x, succ(y)) \to minus(pred(x), y)$

Bei den ersten 3 Regeln gilt schon $l \succ_{emb} r$.
$minus(x, s(y)) \not\succ_{lpo} minus(pred(x), y)$, denn $x \not\succ_{lpo} pred(x)$
(stattdessen: $pred(x) \succ_{lpo} x$)
Lösung: im lexikalischen Vergleich sollte man hier nicht von links nach rechts, sondern von rechts nach links vorgehen.

⇒ Ändere 3. Punkt der lpo wie folgt:
$s = f(s_1, \ldots, s_{i-1}s_i, s_{i+1}, \ldots, s_n)$, $t = f(t_1, \ldots, t_i, s_{i+1}, \ldots, s_n)$, $s_i \succ_{lpo} t_j$ für alle j.
Jetzt $minus(x, s(y)) \succ_{lpo} minus(p(x), y)$, denn $s(y) \succ_{lpo} p(x)$ (bei $minus]p$)
Bsp.: Kombiniere die pred-, minus-Regeln von oben mit den sum-Regeln (vgl. Folie)
Problem: Für sum müssen die Argumente von links nach rechts verglichen werden,
Für minus müssen die Argumente von rechts nach links verglichen werden.
Lösung: Zu jedem n-stelligen Funktionssymbol gibt man einen Status π an, der sagt, in welcher Reihenfolge die Argumente des Funktionssymbols verglichen werden sollen. π ist Permutation der Zahlen $1, \ldots, n$
im Bsp.: sum bekommt Status <1,2>
minus bekommt Status <2,1>
⇒ Lexikalische Pfadordnung mit Status (lpos)
(Terminierung mit lpos bleibt unentscheidbar und NP-Vollständig) ↷ es existieren immer noch TESe, bei denen Terminierung nicht mit lpos gezeigt werden kann.

Beispiel 4.4.10:
$plus(\mathcal{O}, y) \to y$
$plus(s(x), y) \to s(plus(y, x))$
Bei lpos ist die Terminierung nicht zu zeigen, denn $s(x) \not\succ_{lpos} y$ und $y \not\succ_{lpos} x$
Lösung: Man sollte Tupel von Termen so vergleichen, dass die Reihenfolge der Tupelkomponenten keine Rolle spielt.
⇒ Vergleiche Termtupel als Mengen ($\{y, x\} = \{x, y\}$).
Problem: bei Mengen werden Duplikate eliminiert ($\{x, x\} = \{x\}$)
⇒ Vergleiche Termtupel als Multimengen (= Mengen, in denen Elemente mehrmals auftreten dürfen $\{1, 1, 2\} = \{2, 1, 1\} \neq \{1, 2\}$).

Definition 4.4.11: Multimengen
Sei T eine nicht-leere Menge. Eine ungeordnete endliche Folge von Elementen aus T heisst Multimenge über T.
M(T) = Menge aller (endlichen) Multimengen über T.
Eine Multimenge M ist durch ihre charakteristische Funktion $\#_M$: $T \to \mathbb{N}$ definiert, wobei $\#_M(t)$ die Anzahl der Vorkommen von t in M angibt. Es gilt also $t \in M$ gdw. $\#_M(t) > 0$.
Auf Multimengen werden folgende Operationen definiert:

- $M = N$ gdw $\#_M(t) = \#_N(t)$ für alle $t \in T$
- $M \subseteq N$ gdw $\#_M(t) \leq \#_N(t)$ für alle $t \in T$
- $M \subset N$ gdw $\#_M(t) < \#_N(t)$ für alle $t \in T$ gdw $M \subseteq N$, $M \neq N$
- $M \cup N$ 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 $t \in T$
- $M \backslash N$ ist die Multimenge mit $\#_{M\backslash N}(t) = max(=, \#_M(t) - \#_N(t))$

[29.06.04]

Beispiel 4.4.12:
Variation der lpo: Vergleiche Tupel von Argumenten nicht lexikographisch, sondern als Multimengen 4,5,6,6,7,7
Bei lexikographischen Ordnungen: erweitere eine Relation $\succ$ auf T zu einer Relation $\succ^n_{lex}$ auf T^n. Fundiertheit bleibt erhalten.
Bei Multimengen-Ordnungen: erweitere eine Relation $\succ$ auf T zu einer Relation $\succ_{mul}$ auf $M(T)$. Fundiertheit bleibt erhalten.

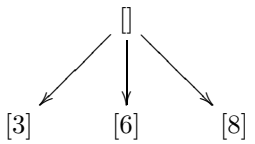
Definition 4.4.13: Multimengenrelation
Sei $\succ$ eine Relation auf T ($\succ \subseteq T \times T$). Dann ist die Relation $\succ_{mul}$ auf $M(T)$ definiert als $M \succ_{mul} N$ gdw

- $X \subseteq M$, $X \neq \emptyset$ (Lasse die Elemente X aus M weg und ersetze sie duch die Elemente Y)
- $N = (M \backslash X) \cup Y$
- für jedes $y \in Y$ existiert ein $x \in X$ mit $x \succ y$

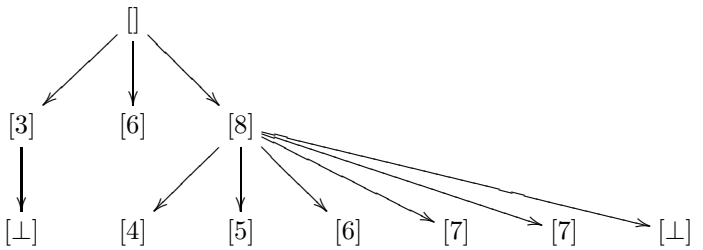
Beispiel 4.4.14:
$M = \{3, 6, 8\}$ $N = \{4, 5, 6, 6, 7, 7\}$: $M(>_{\mathbb{N}})_{mul}N$, denn
$X = \{3, 8\} = M \backslash N$
Idee: Lasse beliebig viele Elemente aus M weg (mindestens eines) und ersetze jedes dieser Elemente durch beliebig viele kleinere (gegebenenfalls durch keines)
$Y = \{4, 5, 6, 7, 7\} = N \backslash M$
für alle $y \in Y$ existiert $8 \in X$ mit $8 >_{\mathbb{N}} y$

Satz 4.4.15: Fundiertheit bleibt bei Multimengen erhalten, Derchowitz 1979
Sei $\succ$ eine Relation auf der Menge T. Dann gilt: $\succ$ ist fundiert gdw $\succ_{mul}$ ist fundiert.
Beweis:
"$\Leftarrow$": Sei $\succ_{mul}$ fundiert.
Ann.: es existiert $t_0, t_1, \ldots \in T$ mit $t_0 \succ t_1 \succ ...$
$\curvearrowright \{t_0\} \succ_{mul} \{t_1\} \succ_{mul} \{t_2\} \succ_{mul} \ldots$
$\lightning$ zur Fundiertheit von $\succ_{mul}$
"$\Rightarrow$": Sei $\succ$ fundiert.
Ann.: es existiert $M_0, M_1, \ldots \in M(T)$ mit $\{M_0\} \succ_{mul} \{M_1\} \succ_{mul} \{M_2\} \succ_{mul} \ldots$
Konstruiere Schritt für Schritt einen unendlichen Baum, dessen Knoten mit Elementen aus T markiert sind.

Baum hat endlichen Verzweigungsgrad und entlang jedes Pfades werden die Elemente aus T bzgl. $\succ$ kleiner.
Lemma von König $\rightarrow$ Baum muss unendlichen Pfad haben $\curvearrowright$
$\lightning$ zur Fundiertheit von $\succ$
Konstruktion des Baumes: erweitere T um ein weiteres Element $\bot$ mit $t \succ \bot$ für alle $t \in T$. Fundiertheit von $\succ$ bleibt erhalten.
Im i-ten Schritt der Konstruktion sind die Blätter des Baums mit den Elementen von M_i markiert.
0. Schritt: beliebige Wurzel, Kinder der Wurzel sind Elemente von M_0 (Bsp.: $M_0 = \{3, 6, 8\}$



)
1. Schritt $M_0 \succ_{mul} M_1 \curvearrowright$ es existiert $X \subseteq M_0$, $X \neq \emptyset$, $M_1 = (M_0 \backslash X) \cup Y$ für alle $y \in Y$ existiert $x \in X$ mit $x \succ y$.
Für jedes $y \in Y$ fügen wir ein neues Blatt hinzu und machen daraus das Kind eines Knoten $x \in X$ mit $x \succ y$.
Ausserdem erhält jeder Knoten $x \in X$ ein Kind $\bot$.
$\curvearrowright$ Blätter sind die Elemente von M_1 (und $\bot$)



(Bsp.: $M_1 = \{4, 5, 6, 6, 7, 7\}$ $X = \{3, 8\}$ $Y = \{4, 5, 6, 7, 7\}$)
$i + 1$ - ter Schritt: analog
$\rightarrow$ Verzweigungsgrad bleibt endlich
$\rightarrow$ entlang eines Pfades werden die Knoten immer kleiner bzgl. $\succ$ (*)
$\rightarrow$ in jedem Schritt wächst der Baum $\curvearrowright$ Baum wird unendlich $\overset{Lemma von König}{\curvearrowright}$
es existiert unendlicher Pfad. Mit (*) ergibt sich $\lightning$ zur Fundiertheit von $\succ$
$\square$

Definition 4.4.16: Rekursive Pfadordnung, Derchowitz 1982
Für eine Signatur Σ und eine fundierte Ordnung] über Σ ("Präzedenz") ist die rekursive Pfadordnung (rpo) über $\mathcal{T}(\Sigma, \mathcal{V})$ definiert als $s \succ_{rpo} t$ gdw

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

- $s = f(s_1, \ldots, s_n)$, $t = g(t_1, \ldots, t_n)$, $f]g$ und $s \succ_{rpo} t_j$ für alle $j \in \{1, \ldots, m\}$
- $s = f(s_1, \ldots, s_n)$, $t = f(t_1, \ldots, t_n)$ und $\{s_1, \ldots, s_n\}(\succ_{rpo})_{mul}\{t_1, \ldots, t_n\}$

Beispiel 4.4.17:

$$plus(\mathcal{O}, y) \rightarrow y plus(\mathcal{O}, y) \succ_{rpo} y$$
$$plus(s(x), y) \rightarrow s(plus(y, x)) plus(s(x), y) \succ_{rpo} s(plus(y, x)), \text{ denn bei } plus]s$$
$$plus(s(x), y) \succ_{rpo} plus(y, x), \text{ denn } \{s(x), y\}(\succ_{rpo})_{mul}\{y, x\}$$
$$\{s(x), y\}(\succ_{rpo})_{mul}\{y, \underline{x}\}, \text{ denn } s(x) \succ_{rpo} x$$

(Genauso: $\{\underline{s(x)}, y\}(\succ_{rpo})_{mul}\{\underline{x}, \underline{x}\}$
$\curvearrowright plus(s(x), y) \succ_{rpo} plus(x, x))$

Satz 4.4.18: Die rekursive Pfadordnung ist eine Simplifikationsordnung
Es existieren Beispiele, bei denen rpo erfolgreich ist und lpos scheitert.
Es existieren Beispiele, bei denen lpos erfolgreich ist und rpo scheitert.
Bei sum: $sum(s(x), y) \not\succ_{rpo} sum(x, s(y))$, denn $\{s(x), y\}(\not\succ_{rpo})_{mul}\{\underline{x}, \underline{s(y)}\}$, denn $s(x) \not\succ_{rpo} s(y)$, $y \not\succ_{rpo} s(y)$
Vorteil rpo: Reihenfolge der Argumente ist unerheblich.
Vorteil lpo: Man kann in manchen Argumenten kleiner und in anderen grösser werden.
Wie erhält man eine Ordnung, die die Vorteile von rpo und lpos verbindet?
⇒ Erweitere das Konzept des Status: Jedes Funktionssymbol bekommt einen Status: Entweder Permutation der Zahlen 1, ..., n (bei n-stellingen Funktionssymbol) $\curvearrowright$ dann muss man die Argumente lexikographisch wie bei lpos vergleichen. Oder Status ist "mul" $\curvearrowright$ vergleiche Argumente als Multimenge wie bei rpo.
⇒ rekursive Pfadordnung mit Status (rpos)
Zeigt die Terminierung des sum + plus - TES, wenn sum Status <1,2> und plus Status "mul" bekommt.
Es ist entscheidbar, ob es zu einem TES eine rpos gibt, mit der man die Terminierung zeigen kann.
Es existieren immer noch terminierendes TESe, deren Terminierung nicht mit rpos gezeigt werden kann.
→ Andere Simplifikationsordnungnen (Knuth-Bendix Ordnung, Polynomordnungen)
→ Es existieren auch terminierende TESe, deren Terminierung mit keiner Simplifikationsordnung gezeigt werden kann.
Bsp: $f(f(x)) \rightarrow f(g(f(x)))$
Terminiert, denn in jedem Reduktionsschritt verringert sich die Zahl der nebeneinander stehenden f's.
Aber: $f(g(f(x))) \succ_{emb} f(f(x))$, d.h. $f(g(f(x))) \succ f(f(x))$ in jeder Simplifikationsordnung $\succ$

$\curvearrowright f(f(x)) \not\succ f(g(f(x))))$, wegen Fundiertheit von $\succ$
Es existieren unendlich viele natürliche TESe / Programme mit demselben Problem
$\rightarrow$ Weitergehende Termmethoden, die auf Simplifikationsordnungen aufbauen (z.B Dependency Pairs)
Techniken für Terminierungsbeweise von TESen können für Terminierungsnachweis von Programmen in verschiedenen Programmiersprachen angepasst werden.

[02.07.04]

Kapitel 5

Konfluenz von Termersetzungssystemen

Generell interessant: Techniken zum Nachweis der Eindeutigkeit von Programmen
Beruht auf Unifikation

- Konfluenzprüfung und Vervollständigung von TESen
- Haupttechnik zur Auswertung von Logikprogrammen
- Typüberprüfung in polymorph getypten Programmiersprachen (z.B. die meisten funktionalen Sprachen: Haskell, ML)
- Automatisches Beweisen (z.B. Resolutionskalkül)

5.1 Unifikation

Definition 5.1.1: Unifikation
Zwei Terme s, t sind unifizierbar gdw. es eine Substitution σ gibt, so dass $s\sigma = t\sigma$. Solch eine Substitution heisst Unifikator von s und t.
Bsp.: $g(f(x), y) =^? g(y, f(z))$ ist lösbar. Unifikator $\sigma = \{x/z, y/f(z)\}$
Ein Unifikationsproblem über $\Sigma, \mathcal{V}$ ist eine endliche Menge von Termgleichungen $S = \{s_1 =^? t_1, \ldots, s_n =^? t_n\}$ mit $s, t \in \mathcal{T}(\Sigma, \mathcal{V})$. Eine Substitution σ ist Lösung oder Unifikator des Unifikationsproblems S gdw. $s_i\sigma = t_i\sigma$ für alle $i \in \{1, \ldots, n\}$.
U(S) ist die Menge aller Unifikatoren von S.
Im allgemeinen gibt es mehrere Unifikatoren (meist unendlich viele):
Im obigen Beispiel sind neben σ auch die folgenden Substitutionen Unifikatoren:
$\sigma_1 = \{x/a, y/f(a), z/a\} \quad (a \in \Sigma_0)$
$\sigma_1 = \{x/f(z), y/f(f(z)), z/f(z)\}$
$\sigma_1 = \{z/x, y/f(x)\}$

Die Substitutionen σ und σ_3 sind allgemeiner als die Substitutionen σ_1 und σ_2: Man kann $\sigma_1, \sigma 2$ erhalten, indem man erst σ oder σ_3 anwendet und anschliessend eine weitere Substitution anwendet.
$\sigma_1 = \underbrace{\sigma \circ \delta_1}_{\text{Komposition: erst } \sigma \text{ anwenden, dann } \delta_1}$ mit $\delta_1 = \{z/a\}$, $\sigma_2 = \sigma \circ \delta_2$ mit $\delta_2 = \{z/f(z)\}$, $\sigma_3 = \sigma \circ \delta_3$ mit $\delta_3 = \{z/x\}$
$\sigma_1 = \sigma \circ \delta_1'$ mit $\delta_1' = \{x/a\}$, $\sigma_2 = \sigma \circ \delta_2'$ mit $\delta_2' = \{x/f(z)\}$, $\sigma_3 = \sigma \circ \delta_3'$ mit $\delta_3' = \{x/z\}$
$\Rightarrow \sigma$ und σ_3 sind allgemeiner als σ_1, σ_2. $\Rightarrow$ sind allgemeine Unifikatoren.

Definition 5.1.2: Allgemeinster Unifikator
Eine Substitution σ ist allgemeiner als σ' gdw. es eine Substitution δ mit $\sigma' = \sigma \circ \delta$.
Eine Substitution σ ist allgemeinster Unifikator (most general unifier, mgu) eines Unifikationsproblems S gdw. $\sigma \in U(S)$ und σ allgemeiner als alle $\sigma' \in U(S)$ ist.
Schreibweise: "$\sigma \circ \delta$" wird oft als "$\sigma\delta$" geschrieben.
$\curvearrowright t(\sigma \circ \delta = t\sigma\delta = (t\sigma)\delta$
Bsp.: verschiedene mgu's:
$\sigma = \{x/z, y/f(z)\}$
$\sigma_3 = \{z/x, y/f(x)\}$
Diese beiden Substitutionen sind "gleich allgemein" $\curvearrowright$ sind identisch bis auf Variablenumbenennung

Lemma 5.1.3: Eindeutigkeit allgemeinster Unifikatioren
Seien σ, σ' zwei Substitutionen. Falls σ allgemeiner als σ' ist und σ' allgemeiner als σ ist, dann existiert eine Variablenumbenennung δ mit $\sigma' = \sigma \circ \delta$. Hierbei heisst eine Substitution δ Variablenumbennenung gdw. δ injektiv ist und $x\delta \in \mathcal{V}$ für alle $x \in \mathcal{V}$ gilt.
Beweis: Übung
$\sigma_3 = \sigma \circ \delta$, δ Variablenumbenennung. $\delta_3 = \{z/x\}$ ist nicht injektiv $\delta_3(z) = \delta_3(x) = x$
$\delta = \{z/x, x/z\}$
Mit einem einzigem mgu kann man die Menge der Lösungen U(S) repräsentieren.

Lemma 5.1.4: Darstellung der Lösungen eines Unifikationsproblems
Sei S ein Unifikationsproblem über $\Sigma, \mathcal{V}$, sei σ ein allgemeinster Unifikator von S. Dann gilt:
$U(S) = \{\sigma\delta | \delta \in \text{SUB}(\Sigma, \mathcal{V})\}$
Beweis:
"$\subseteq$": $\sigma' \in U(S)$.
Dann gilt $\sigma' \in \{\sigma\delta | \delta \in \text{SUB}(\Sigma, \mathcal{V})\}$, nach der Definition des mgu.
"$\supseteq$": Sei $\delta \in \text{SUB}(\Sigma, \mathcal{V})$.
Dann gilt $\sigma\delta \in U(S)$, denn für alle $s =^? t \in S$ gilt:
$s\sigma = t\sigma$ (denn $\sigma \in U(S)$)

$\curvearrowright (S\sigma)\delta = (t\sigma)\delta$.
□

Ziel: Entwickle Unifikationsalgorithmus, der zu einem Unifikationsproblem S entscheidet, ob es lösbar ist und falls ja, einen mgu berechnet.
Vorgehen: Transformiere das Unifikationsproblem S so lange, bis es in "gelöster Form" ist. Dann kann man den mgu direkt ablesen.

Definition 5.1.5:
Ein Unifikationsproblem $S = \{x_1 =^? t_1, \ldots, x_n =^? t_n\}$ ist in gelöster Form gdw. die x_i paarweise verschiedene Variablen sind, die in den Termen $t_1, \ldots, t_n$ nicht vorkommen. Zu einem Unifikationsproblem in gelöster Form wie oben definieren wir die Substitution $\sigma_S = \{x_a/t_1, \ldots, x_n/t_n\}$.
Bsp.: $\{g(f(x), y) =^? g(y, f(z))\} \Rightarrow^* \{x =^? z, y =^? f(z)\}$: $\sigma_{S'} = \{x/z, y/f(z)\}$
$\{x =^? f(x)\}$ ist nicht lösbar: Für jede Substitution σ hat $f(x)\sigma$ ein Zeichen mehr als $x\sigma \curvearrowright f(x)\sigma \neq x\sigma$.
$x =^? t$ mit $x \in \mathcal{V}(t)$ ist imer unlösbar, falls $t = x$. (Occur Failure)

Lemma 5.1.6:
Sei S ein Unifikationsproblem in gelöster Form. Dann ist σ_S mgu von S und für alle Unifikatoren σ' von S gilt: $\sigma' = \sigma_S \circ \sigma'$.
Beweis:

- $\sigma_s \in U(S)$: Für alle $x_i =^? t_i \in S$ gilt: $x_i\sigma_S = t_i$ $t_i\sigma_S \underbrace{=}_{\text{denn die Variablen } x_1, \ldots, x_n \text{ treten nicht in } t_i \text{ auf}} t_i\sigma_S$
- für alle $\sigma' \in U(S)$. $\sigma' = \sigma_S\sigma'$
  $\curvearrowright$ Zeige $y\sigma' = y\sigma_s\sigma'$ für alle Variablen y.

Fall 1 $y \in \{x_1, \ldots, x_n\}$
$x_i\sigma' \underbrace{=}_{\text{denn } \sigma' \in U(S)} t_i\sigma' = \underbrace{x_i\sigma_S\sigma'}_{t_i}$

Fall 2 $y \notin \{x_1, \ldots, x_n\}$
$y\sigma' = \underbrace{Y\sigma_S\sigma'}_{y}$

Definition 5.1.7: Transformation von Unifikationsproblemen
Wir definieren folgende Relation $\Rightarrow$ auf Unifikationsproblemen durch 4 Transformationsregeln:

- Löschen: $S \underbrace{\uplus}_{\text{disjunkte Vereinigung}} \{t =^? t\} \Rightarrow S$
- Termreduktion: $S \uplus \{f(s_1, \ldots, s_n) =^? f(t_1, \ldots, t_n)\} \Rightarrow S \cup \{s_1 =^? t_1, \ldots, s_n =^? t_n\}$

- Vertauschen: $S \uplus \{t =^? x\} \Rightarrow S \cup \{x =^? t\}$, falls $t \notin \mathcal{V}$
- Variablenreduktion: $S \uplus \{x =^? t\} \Rightarrow \{u\sigma =^? v\sigma$ für die gilt $u =^? v \in S\} \cup \{x =^? t\}$, falls $\sigma = \{x/t\}$, $x \notin \mathcal{V}(t)$, $x \in \mathcal{V}(S)$

[09.07.04]

- mgu ist bis auf Variablenumbennenung eindeutig
- Falls σ mgu von S ist, dann $U(S) = \{\sigma\delta | \delta \in \mathrm{SUB}(\sigma, \mathcal{V})\}$
- lösen von Unifikationsproblemen $S \Rightarrow S_1 \Rightarrow S_2 \Rightarrow \ldots \Rightarrow S_n$
- Unifikationsproblem S in glöster Form: $S = \{x_1 =^? t_k\}$ $x_1, \ldots, x_k$ paarweise verschieden, $X_i \notin \mathcal{V}(t_0)$ für alle i, j mgu ist $\sigma_S = \{x_1/t_1, \ldots, x_k/t_k\}$

Beispiel 5.1.8: Folie
Bsp. für unlösbare Unifikationsprobleme
$\{x =^? f(x)\}$, denn $x\sigma \neq f(x\sigma)$ für alle Substitutionen σ OCCUR FAILURE
$\{g(x) =^? g(x)\}$, denn $f(x\sigma) \neq g(y\sigma)$ für alle Substitutionen σ CLASH FAILURE

Unifikationsalgorithmus UNIFY
(Robinson 1969, Martell und Montenau 1982 $\widehat{=}$ unserer Formulierung)
Eingabe: Unifikationsproblem S
Ausgabe: mgu von S, falls S lösbar ist, sonst "False"
(es existieren Unifikationsalgorithmen mit linearem Aufwand (unserer ist exponentiell))

Satz 5.1.9: Korrektheit des Unifikationsalgorithmus
Sei S ein Unifikationsproblem

a) Die Relation $\Rightarrow$ ist fundiert ($\curvearrowright$ UNIFY terminiert)

b) Falls $S \Rightarrow S'$, dann gilt $U(S) = U(S')$ (Menge der Unifikatoren des letzten Problems sind die Unifikatoren des ersten Problems)

c) Falls S lösbar ist und in Normalform bzgl. $\Rightarrow$ ist, dann ist S in gelöster Form (Korrektheit von Schritt 2 von UNIFY)

d) Der Algorithmus UNIFY terminiert und ist korrekt

Beweis:

1. Ziel: Finde ein Maß, so dass Unifikationsproblem bei jedem Transformationsschritt kleiner wird

   Var x ist gelöst gdw. sie genau einmal in S auftritt und zwar in einer Gleichung $x =^? t$ $(x \notin \mathcal{V}(t))$

   Messe drei Maße für Unifikationsprobleme:

	n_1	n_2	n_3
$\{g(f(a), g(x,x)) =^? g(x, g(x,y))\} \Rightarrow_{\text{Termred}}$	2	11	0
$\{f(a) = x, g(x,x) =^? g(x,y)\} \Rightarrow_{\text{vertauschen}}$	2	9	1
$\{x =^? f(a), g(x,x) =^? g(x,y)\} \Rightarrow_{\text{Variablenred}}$	2	9	0
$\{x =^? f(a), g(f(a), f(a)) =^? g(f(a), y)\} \Rightarrow_{\text{Termred}}$	1	12	0
$\{x =^? f(a), f(a) =^? f(a), f(a) =^? x\} \Rightarrow_{\text{Löschen}}$	1	10	1
$\{x =^? f(a), f(a) =^? y\}$	1	6	1

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

- n_1 = Anzahl der Variable in S, die noch nicht gelöst sind
- n_2 = Anzahl der Zeichen in S, d.h. $n_2 = \Sigma_{S=^? t \in S}|S| + |t|$
- n_3 = Anzahl der Gleichungen $t =^? x$ in S mit $t \notin \mathcal{V}$

Maß für Unifikationsprobleme ist (n_1, n_2, n_3). Dieser 3-Tupel wird bei jedem Transformationsschritt bzgl. $(>_{\mathbb{N}})^3_{lex}$ kleiner

[float Box]

Generell

$\{f(x) =^? f(a)\} \Rightarrow \{x =^? a\}$

$n_1 = 1$ $n_1 = 0$

$\{a =^? x\} \Rightarrow$

.....

[end box]

2. $S \Rightarrow S' \curvearrowright U(S) = U(S')$. Trivial für die ersten 3 Regeln. Zeige es für Variablenreduktion

   $S \uplus \{x =^? t\} \Rightarrow \{u\sigma =^? u \in S\} \cup \{x =^? t\}$, wobei $\sigma = \{x/t\}$

   σ ist mgu von $\{x =^? t\}$ und für alle Θ mit $x\Theta = t\Theta$, gilt $\Theta = \sigma\Theta$ (Lemma 5.1.6, da $\{x =^? t\}$ in gelöster Form)

   $\Theta \in U(S \uplus \{x =^? t\})$ gdw $x\Theta = t\Theta$ und $\Theta \in U(S)$

   gdw $x\Theta = t\Theta$ und $\sigma\Theta \in U(S)$ (wegen $\Theta = \sigma\Theta$)

   gdw $x\Theta = t\Theta$ und $u\sigma\Theta = v\sigma\Theta$ für alle $u =^? v \in S$

   gdw $x\Theta = t\Theta$ und $\Theta \in U\{u\sigma =^? v\sigma | u =^? v \in S\})$

   gdw $\Theta \in U(\{u\sigma =^? v\sigma | u =^? v \in S\} \cup \{x =^? t\})$

3. S lösbar $\curvearrowright$ S kann keine Gleichung $f() =^? g()$ enthalten

   S in Normalform $\curvearrowright$ S kann keine Gleichung $f() =^? f()$ enthalten (sonst wäre Termreduktion anwendbar)

   Aus $s =^? t \in S$ folgt also $S \in \mathcal{V}$ oder $t \in \mathcal{V}$

   Wäre $S \notin \mathcal{V}$, dann wäre „Vertauschen“ anwendbar

   $\curvearrowright$ Aus $s =^? t \in S$ folgt also $s \in \mathcal{V}$. S enthält also nur Gleichungen der Form $x =^? t$. Hierbei $x \notin \mathcal{V}(t)$

   - $p = x$ unmöglich da sonst Löschen anwendbar
   - $x \in \mathcal{V}(t)$, $x \neq t$ das sonst OCCUR FAILURE, S unlösbar

   x tritt auch nirgendwo sonst in S auf (ist gelöst), da sonst „Variablenreduktion“ anwendbar wäre.

   $\Rightarrow$ S ist in gelöster Form

4. Terminierung folgt aus (a) $S \Rightarrow S_1 \Rightarrow \ldots \Rightarrow S_n$

   wegen (b) $U(S) = U(S_n)$ (Induktion über n, Länge der Transformation.

   Falls S lösbar $\curvearrowright U(S) = U(S_n) \neq \emptyset \overset{(c)}{\curvearrowright} S_n$ ist in gelöster Form, σ_{S_n} ist mgu von S_n und

   Falls S unlösbar $\curvearrowright U(S) = U(S_n) = \emptyset \overset{(c)}{\curvearrowright} S_n$ ist nicht in gelöster Form $\curvearrowright$ “false“

Korollar 5.1.10: Unifikationssatz
Wenn ein Unifikationsproblem lösbar ist, dann hat es einen mgu

[13.07.04]

Effizienz von UNIFY kann verbessert werden, in dem man z.B. abbricht (mit “False“), sobald Gleichungen auftreten, die zur unlösbarkeit des Unifikationsproblems führen: $f(\ldots) =^? g(\ldots)$ Clash Failure oder $x =^? t$, $x \in \mathcal{V}(t)$, $x \neq t$ Occur Failure
Unifikation: s unifiziert t gdw. es existiert σ mit $s\sigma = t\sigma$ (z.B. nötig, um logische Programmiersprachen zu implementieren)
Matching: s matcht t gdw. es existiert σ mit $s\sigma = t$ (z.B. nötig, um Termersetzung oder funktionale Programmeirsprachen zu implementieren)
Idee für einen Matching Algorithmus: Rufe UNIFY mit s und $t\Theta$ auf, wobei Θ alle Variablen $x \in \mathcal{V}(t)$ durch neue Konstanten c_x ersetzt. Falls σ der mgu von s und $t\Theta$ ist, so erhält man den Matcher von s und t, in dem man in σ alle c_x wieder durch x ersetzt.
Algorithmus MATCH(s, t)
Eingabe: 2 Terme s, t
Ausgabe: Matcher σ mit $s\sigma = t$, falls er existiert. Sonst “False“.

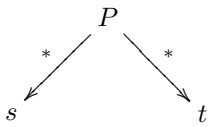
1. Sei $\Theta = \{x/c_x | x \in \mathcal{V}(T)\}$, wobei c_x neue Konstanten sind.

2. Berechne UNIFY$(s, t\Theta)$
3. Falls UNIFY "False" liefert, dann brich ab mit "False"
4. Falls UNIFY $\{x_1/t_1, \ldots, x_n/t_n\}$ liefert, so gib aus $\{x_1/t'_1, \ldots, x_n/t'_n\}$, wobei t'_i aus t_i entsteht, indem die Konstanten c_x wieder durch x ersetzt werden.

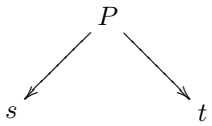
Beispiel 5.1.11:
Berechne Matcher von $s = g(x, y)$ und $t = g(f(x), x)$. (s, t sind nicht unifizierbar)
$\Theta = \{x/c_x\}$. UNIFY($\underbrace{g(x, y)}_{s}, \underbrace{g(f(c_x), c_x)}_{t\Theta}$) liefert mgu $\{x/f(c_x), y/c_x\}$.
Matcher ist {x/f(x), y/x}

5.2 Lokale Konfluenz und Kritische Paare

Im Allgemeinen: Konfluenz von TESen ist unentscheidbar
(aber: bei terminierenden TESen ist Konfluenz entscheidbar)
Vorgehen: Weise nicht Konfluenz nach, sonder eine schwächere Eigenschaft: lokale Konfluenz
Konfluenz untersucht Indeterminismen nach beliebig vielen Schritten:



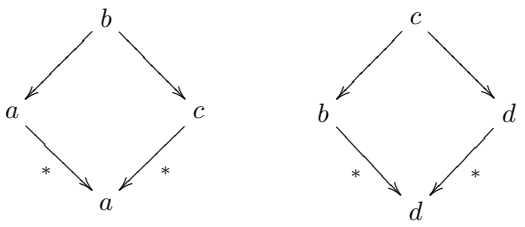
Lokale Konfluenz untersucht Indeterminismen nach einem Schritt:



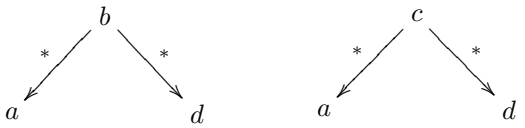
Definition 5.2.1:
Eine Relation $\rightarrow$ auf einer Menge M ist lokal Konfluent gdw. für alle $p, s, t \in M$ gilt: Falls $p \rightarrow s$ und $p \rightarrow t$ gilt, dann existiert ein $q \in M$ mit $s \rightarrow^* q$, $t \rightarrow^* q$

Lokale Konfluenz $\leftarrow$ Konfluenz
Aber: Lokale Konfluenz $\not\leftarrow$ Konfluenz

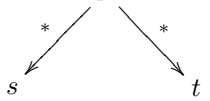
Beispiel 5.2.2:
$\mathcal{R} = \{b \rightarrow a, b \rightarrow c, c \rightarrow b, c \rightarrow d\}$
Lokale Konfluenz:

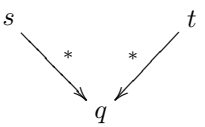


$\mathcal{R}$ ist lokal konfluent
Konfluenz:



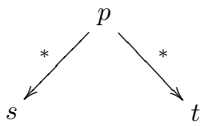
$\mathcal{R}$ ist nicht konfluent
Aber bei terminierenden TESen: lokale Konfluenz → Konfluenz

Satz 5.2.3: Diamond Lemma, Satz von Newman, 1942
Sei → eine fundierte Relation über einer Menge M. Dann ist → konfluent gdw. sie lokal konfluent ist.
Beweis:
"⇒": trivial
"⇐": Vorraussetzung: → ist lokal Konfluent. Zu zeigen ist die folgende Aussage für alle $p \in M$:
Falls p 

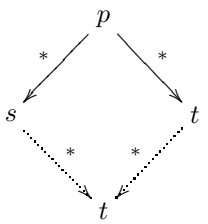
, dann $s \downarrow t$. (d.h. es existiert q mit 

) (*)

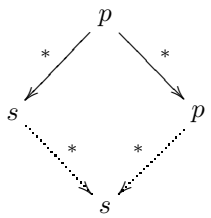
Zeige (*) durch noethersche Induktion über p und verwenden → als Induktionsrelation. ⇐ nur möglich, weil "→" fundiert ist
Für alle p' mit $p \to p'$ können wir (*) als Induktionshypothese voraussetzen.
Sei



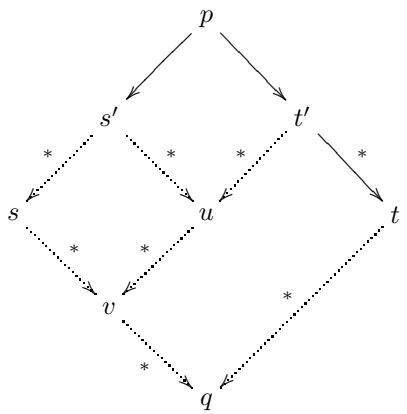
Fall 1 $p = s$

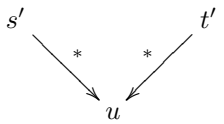


Fall 2 $p = t$

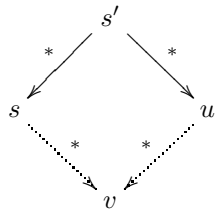


Fall 3 $p \neq s, p \neq t$ Man braucht also mindestens einen Schritt von p zu s und zu t

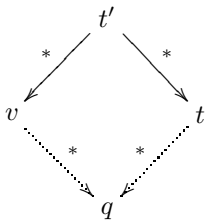




weil $\rightarrow$ lokal konfluent ist



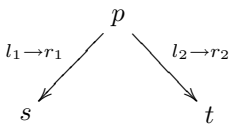
weil t' kleiner als p ist bzgl. Induktionsrelation und man daher die Induktionshypothese (*) für s' benutzen darf



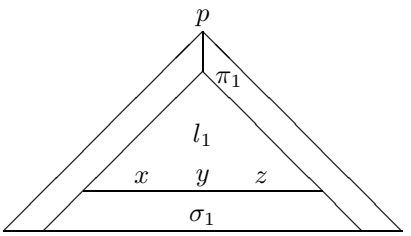
weil t' kleiner als p ist bzgl. Induktionsrelation und man daher die Induktionshypothese (*) für t' benutzen darf

□

Bei terminierenden TESen: Konfluenztest kann lokalisiert werden – betrachte nur noch Indeterminismen nach 1 Schritt.
(Lokale) Konfluenz ist bei terminierenden TESen entscheidbar: die meisten dieser Indeterminsimen sind immer zusammenführbar.
Es gibt nur endlich viele Indeterminismen, die man tatsächlich für lokale Konfluenz untersuchen muss.
Vorgehen: Untersuche alle kritischen Situationen der Art:

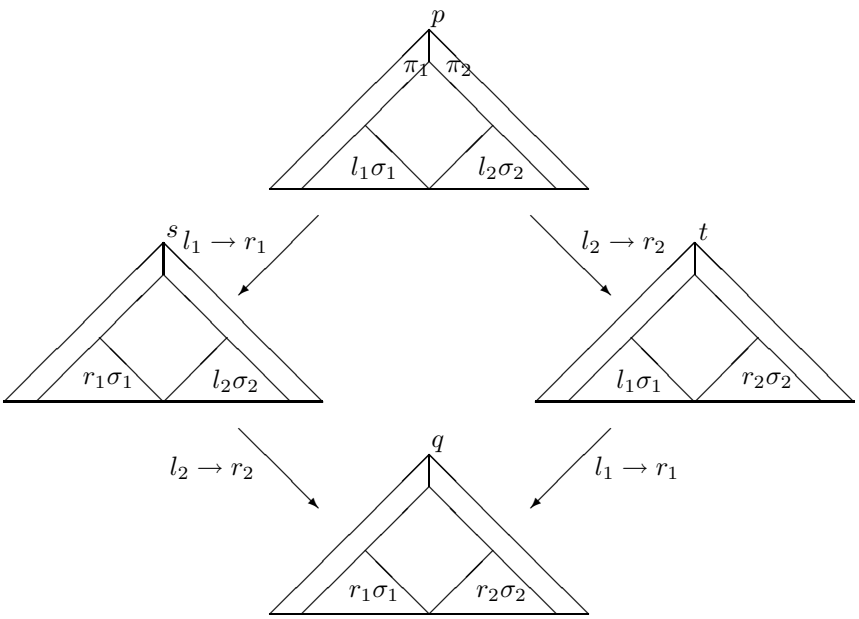


Analysiere an welchen Stellen von p die Regeln angewendet wurden. $l_1 \to r_1$ wird an Stelle π_1 angewendet, $l_1 \to r_2$ wird an Stelle π_2 angewendet.
p hat also folgende Gestalt:



$P|_{\Pi_1} = l_1\sigma_1$ und $s = p[r_1\sigma_1]_{\Pi_1}$
$P|_{\Pi_2} = l_2\sigma_2$ und $s = p[r_2\sigma_2]_{\Pi_2}$
Betrachte, wie Π_1 und Π_2 zueinander liegen und ob durch die Reduktion an Π_1 der Redex an Π_2 zerstört werden kann (u. u.)
Nur dann sind s, t evtl nicht mehr zusammnführbar.

Fall 1 $\Pi_1 \perp \Pi_2$ (d.h. $\Pi_2 \not\geq_{\mathbb{N}^*} \Pi_1$, $\Pi_1 \not\geq_{\mathbb{N}^*} \Pi_2$)



$p = p[l_1\sigma_1]_{\pi_1}[l_2\sigma_2]_{\pi_2}$

$s = p[r_1\sigma_1]_{\pi_1}[l_2\sigma_2]_{\pi_2}$

$t = p[l_1\sigma_1]_{\pi_1}[r_2\sigma_2]_{\pi_2}$

$s \to q$, $t \to q$ mit $q = p[r_1\sigma_1]_{\pi_1}[r_1\sigma_2]_{\pi_2}$

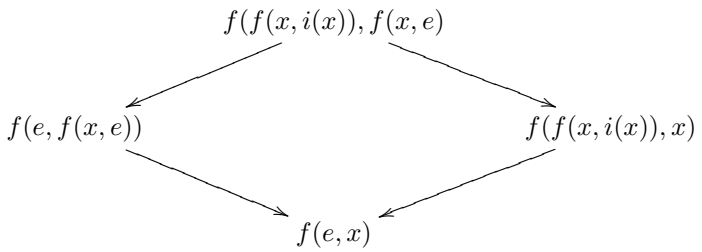
s und t lassen sich also in 1 Schritt zusammenführen. $\Rightarrow$ muss man nicht untersuchen.

Bsp.:

$f(x, f(y, z))) \to f(f(x, y), z)$

$f(x, e) \to x$

$f(x, i(x)) \to e$

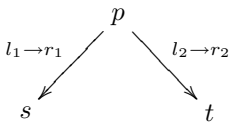


$\pi_1 = 1$

$\pi_2 = 2$

$\pi_1 \downarrow \pi_2$

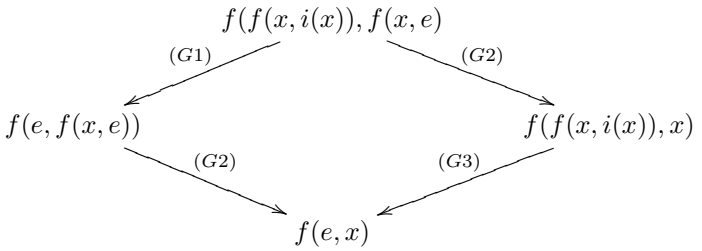
[16.07.04]



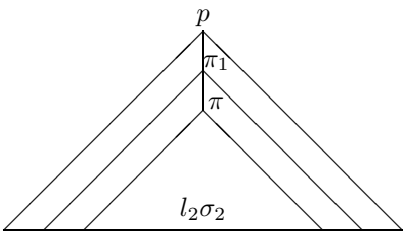
$p|_{\pi_1} = l_1\sigma_1,\ s = p[r_1\sigma_1]_{\pi_1}$
$p|_{\pi_2} = l_2\sigma_2,\ s = p[r_2\sigma_2]_{\pi_2}$
Untersuche die Fälle, wie sich π_1 und π_2 zueinander verhalten.

Fall 1 Dann sind s und t immer zusammenführbar.

$$f(x, f(y, z)) \to f(f(x, y), z) \quad (G1)$$
$$f(x, e) \to x \quad (G2)$$
$$f(x, i(x)) \to e \quad (G3)$$

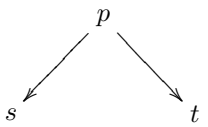


Fall 2 π_1 liegt über π_2 ($\pi_2 \geq_{\mathbb{N}^*} \pi_1$)

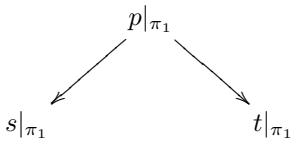


$\pi_2 = \pi_1\pi$ ($\pi = \mathcal{E}$ ist möglich)

Anstelle der kritischen Situation



betrachten wir nun



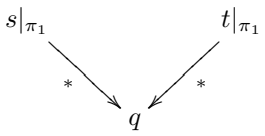
Grund: Wenn $s|_{\pi_1}$ und $t|_{\pi_1}$ zusammenführbar sind, dann auch s und t.

$p|_{\pi_1} = l\sigma_1 = l_1\sigma_1[l_2\sigma_2]_\pi$

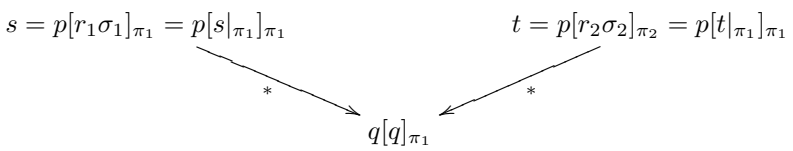
$s|_{\pi_1} = r\sigma_1$

$t|_{\pi_1} = l_a\sigma_1[r_2\sigma_2]_\pi$

Falls

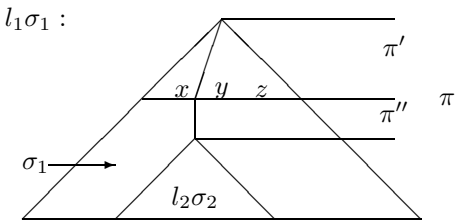


dann



Unterscheide danach, wie "tief" $l_2\sigma_2$ in $l_1\sigma_1$ liegt.

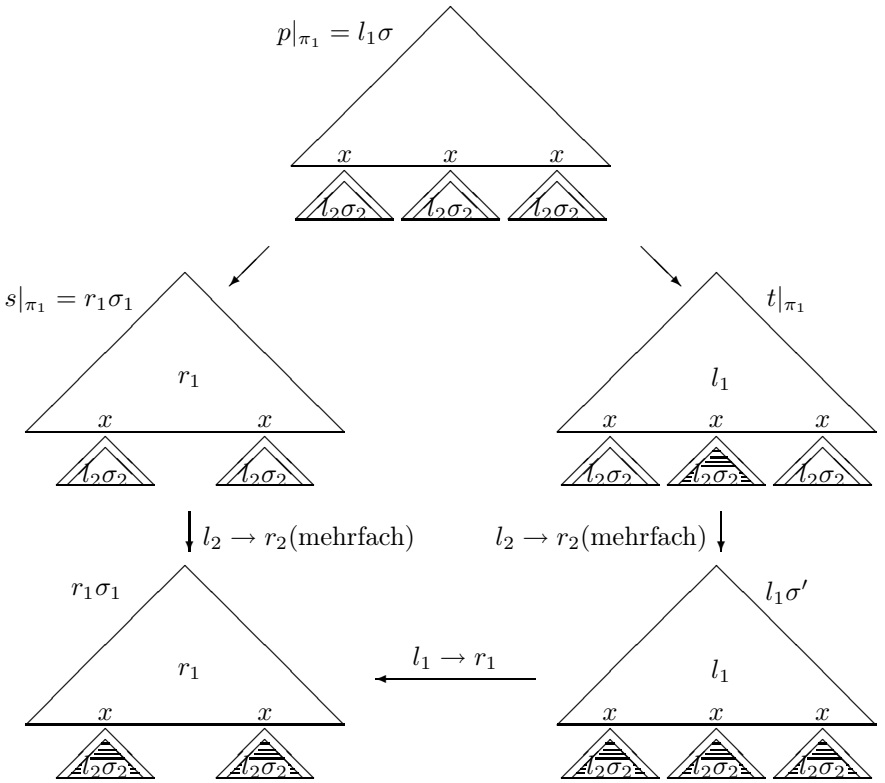
Fall 2.1: $l_2\sigma_2$ liegt „im Subsitutionsteil", d.h. $\pi \notin Occ(l_1)$, $l_1|_{\pi'} = x \in \mathcal{V}$



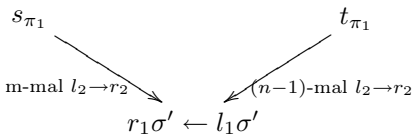
$\pi = \pi'\pi''$ mit $\pi \in Occ(l_1)$, $l_1|_{\pi'} = x \in \mathcal{V}$, $\pi'' = \mathcal{E}$ ist möglich

$l_2\sigma_2 = x\sigma_1|_{\pi''}$

Wenn l_1 die Variable x n-mal enthält, dann tritt $l_2\sigma_2$ n-mal in $l_1\sigma_1$ auf.
$\curvearrowright$ Die Regel $l_2 \to r_2$ lässt sich also n-mal auf $l_1\sigma_1$ anwenden.

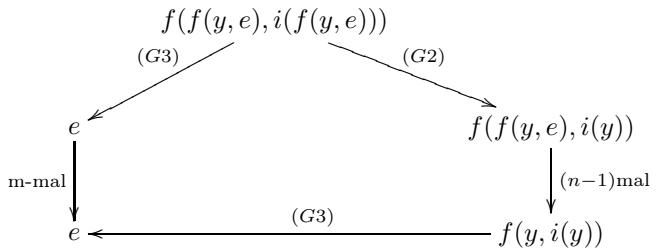


Wenn r_1 die Variable x m-mal enthält, dann tritt $l_2\sigma_2$ m-mal in $r_1\sigma_1$ auf.
Daher:



wobei σ' wie σ ist, aber $x\sigma' = x\sigma[r_1\sigma_2]_{\pi''}$

$\Rightarrow$ Die Indeterminismen im Fall 2.1. Kann man <u>immer</u> zusammenfassen!



$\pi_1 = \mathcal{E}$

$\pi_2 = 2 \quad 1$

$l_1 = \{x/f(y,e)\}$

$l_2\sigma_2 = f(y,e)$

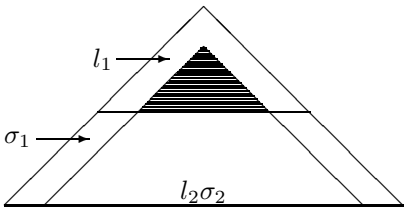
$r_2\sigma_2 = y$

$n = 2$

$m = 0$

$\sigma' = \{x/y\}$

Fall 2.2 $l_2\sigma_2$ liegt nicht „im Substitutionsteil“, d.h. $\pi \in Occ(l_1)$ und $l_1|_\pi \notin \mathcal{V}$



Kritische Überlappung ⇒ dies sind die einzigen Fälle, die man zur Untersuchung der lokalen Konfluenz betrachten muss.

Allgemein: Dieser Fall tritt auf, wenn es 2 Regeln $l_1 \to r_1$, $l_2 \to r_2$ gibt ($l_1 \to r_1 = l_2 \to r_2$ ist möglich) und es existiert $\pi \in Occ(l_1)$ mit $l_1|_\pi \notin \mathcal{V}$ mit $l_1|_\pi\sigma_1 = l_2\sigma_2$.

Wenn wir die Variablen von $l_1 \to r_1$ und $l_2 \to r_2$ so umbenennen, dass sie disjunkt sind, dann muss man untersuchen, ob $l_1|\pi\sigma = l_2\sigma$ gelten kann.

D.h.: Zu jeder linken Seite l_1, l_2 und jedem Nicht-Variablen-Teilterm von l_1 muss man untersuchen: Sind $l_1|_\pi$ und l_2 unifizierbar?

Falls ja, so muss man für den aus $l_1|_\pi\sigma = l_2\sigma$ entstehenden Indeterminismus untersuchen, ob er zusammenführbar ist (σ sei mgu von $l_1|_\pi$ und l_2 sein).

D.h.: untersuche, ob $r_1\sigma$ und $l_1\sigma[r_2\sigma]_\pi$ zusammenführbar sind ⇒ es genügt, die Zusammenführbarkeit der (endlich vielen) kritischen Paare zu überprüfen.

$$l_1\sigma = l_1[l_1|_\pi]_\pi\sigma$$

$$= l_1\sigma[\underbrace{l_1|_\pi\sigma}_{l_2\sigma}]_\pi$$

$$r_1\sigma \qquad\qquad l_1\sigma[r_2\sigma]_\pi$$

Fall 3 „π_2 liegt vor π_1“

analog

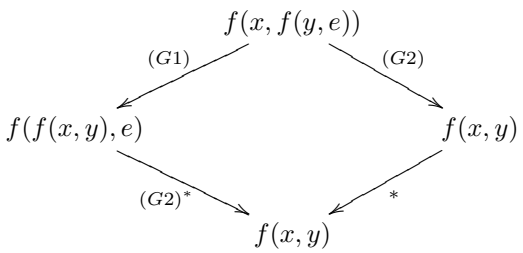
Definition 5.2.4: Kritisches Paar, Knuth + Bendix, 1970

Sei $\mathcal{R}$ ein TES mit $l_1 \to r_1$, $l_2 \to r_2 \in \mathcal{R}$. 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 mgu von $l_1|_\pi$ und l_2. Dann ist $\langle r_1\sigma, l_1\sigma[r_2\sigma]_\pi\rangle$ ein kritisches Paar von $\mathcal{R}$. Die beiden Regeln $l_1 \to r_1$, $l_2 \to r_2$ dürfen bis auf Variablenumbenennung identisch sein. In diesem Fall betrachtet man nur Überlappungen mit $\pi \neq \mathcal{E}$. Die Menge der kritischen Paare von $\mathcal{R}$ wird mit $CP(\mathcal{R})$ bezeichnet.

Es existieren stets nur endlich viele kritische Paare und $CP(\mathcal{R})$ kann mit Hilfe von UNIFY berechnet werden.

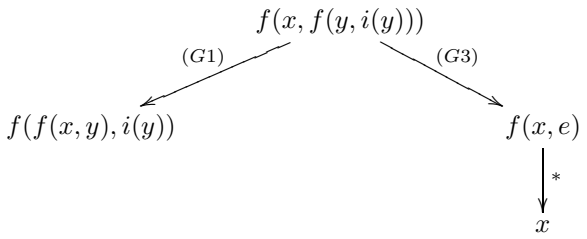
Beispiel 5.2.5: Kritische Paare von {(G1), (G2), (G3)}

- (G1) + (G2): $\underbrace{f(x', e)}_{l_2}$ und $\underbrace{f(y, z)}_{l_1|_\pi}$ unif. mit mgu $\{x'/y, z/e\}$



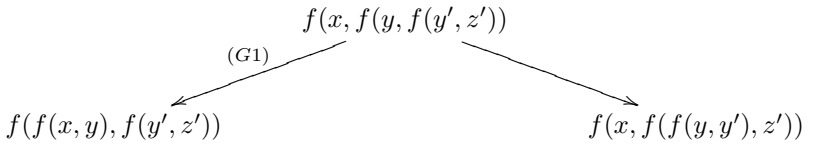
$\langle f(f(x, y), e), f(x, y)\rangle$

- (G1) + (G3): $f(x', i(x'))$ und $f(y, z)$ unif. mit mgu $\{x'/y, z/i(y)\}$



  $\langle f(f(x, y)i(y)), f(x, e)\rangle$
  nicht zusammenführbar $\curvearrowright$ TES ist nicht konfluent

- (G1) + (G1): $f(x', f(y', z'))$ und $f(y, z)$ unif mit mgu $\{x'/y, z/f(y', z')\}$



$\langle \ldots, \ldots \rangle$ ist zusammenführbar

Satz 5.2.6: Kritisches Paar Lemma, Knuth + Bendix, 1970
Sei $\mathcal{R}$ ein TES. $\mathcal{R}$ ist lokal konfluent gdw. alle seine kritischen Paare zusammenführbar sind. (auch wenn $\mathcal{R}$ nicht terminiert)
Beweis:
siehe vorige Überlegungen, alle Indeterminismen, die keinem kritischen Paar entsprechen, sind zusammenführbar
□

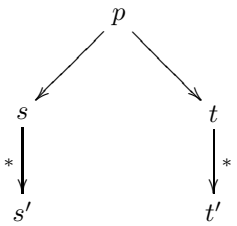
Algorithmus CONFLUENCE($\mathcal{R}$)

1. Berechne $CP(\mathcal{R})$
2. Falls $CP(\mathcal{R}) = \emptyset$, gib aus "True" und brich ab
3. Wähle $\langle s, t\rangle \in CP(\mathcal{R})$
4. Reduziere s, t zu Normalformen s', t'
5. Falls $s' \neq t'$, dann gib aus "False" und brich ab
6. Setze $CP(\mathcal{R}) = CP(\mathcal{R})\{\langle s, t\rangle\}$
7. zurück zu 2

[20.07.04]

[Folie]
Diamond-Lemma: $\mathcal{R}$ lokal konfluent $\Leftrightarrow$ $\mathcal{R}$ konfluent, falls $\mathcal{R}$ terminiert
Überprüfung der Konfluenz bei terminierenden TESen:

1. Bilde alle Kritischen Paare $\langle s, t \rangle$
2. Berechne beliebige Normalformen s' von s, t' von t.
3. $s' = t'$, dann ist das kritische Paar zusammenführbar
4. Falls $s' \neq t'$, dann ist $\mathcal{R}$ nicht konfluent



Satz 5.2.7:

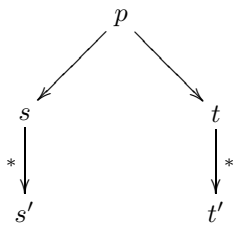
a) Der Algorithmus CONFLUENCE terminiert und ist korrekt

b) Für terminierende TESe ist Konfluenz entscheidbar

Beweis:

a) Terminierung, denn UNIFY terminiert, $|CP(\mathcal{R})|$ ist endlich, Berechnung der Normalformen in Schritt 4 terminiert (weil $\mathcal{R}$ terminiert), bei jedem Durchlauf der Schritte 2-7 wird $CP(\mathcal{R})$ kleiner. Falls der Algorithmus "True" ausgibt $\curvearrowright$ alle kritischen Paare zusammenführbar $\overset{\text{Satz } 5.3.6\ [1]}{\curvearrowright}$ lokale Konfluenz $\overset{\text{Satz } 5.2.3\ [2]}{\curvearrowright}$ Konfluenz

   Falls der Algorithmus "True" ausgibt $\curvearrowright$ so existiert



   d.h. p hat 2 verschiedene Normalformen $\curvearrowright$ $\mathcal{R}$ nicht confluent

b) folgt, da CONFLUENCE ein entscheidungsalgorithmus ist

□

[0]Kritische Paar Lemma
[0]Diamond Lemma

Kapitel 6

Vervollständigung von Termersetzungssystemen

6.1 grundlegender Vervollständigungsalgorithmus, der anschliessend verbessert wird (6.1) Einsatz für Induktionsbeweise/Programmverifikation (6.3)

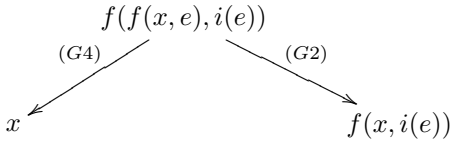
6.1 Der grundlegende Vervollständigungsalgorithmus

Beispiel 6.1.1:
plus-TES: Durch Richten der Gleichungen entsteht ein terminierendes und konfluentes TES $\curvearrowright$ man bekommt direkt ein Verfahren, um Aussagen über plus zu beweise/widerlegen

Beispiel 6.1.2:
Gruppen-TES: Durch Richten der Gleichungen entsteht ein terminierendes TES, das nicht konfluent ist.
Grund: Ein kritisches Paar $\left\langle \underbrace{f(f(x,y),i(y))}_{f(f(x,y),i(y))}, \underbrace{f(x,e)}_{x} \right\rangle$ ist nicht zusammenführbar.
Idee: Bei nicht zusammengeführten kritischen Paaren $\langle s,t\rangle$, wobei s die Normalform s', t die Normalform t' hat, ergänze das TES um die Regel $s' \to t'$ oder $t' \to s'$ (neues TES soll weiterhin terminieren)
$\mathcal{R}_0 = \{(G1),(G2),(G3)\}$ wird vervollständigt zu:
$\mathcal{R}_0 = \{(G1),(G2),(G3), \underbrace{f(f(x,y),i(y)) \to x}_{(G4)}\}$
$\mathcal{R}_1$ ist adäquat für Gruppenaxiome ($\mathcal{E} \subseteq \leftrightarrow^*_{\mathcal{R}_0} \subseteq \leftrightarrow^*_{\mathcal{R}_1}$)

$\mathcal{R}_1$ ist korrekt für Gruppenaxiome ($\mathcal{R}_1 \subseteq \leftrightarrow^*_{\mathcal{E}}$. Wir wissen: $\mathcal{R}_0 \subseteq \leftrightarrow^*_{\mathcal{E}}$. $f(f(x,y)i(y) \underbrace{\leftarrow^*_{\mathcal{R}_0} \ldots \rightarrow^*_{\mathcal{R}_0}}_{\leftrightarrow^*_{\mathcal{R}_0}} x)$

$\mathcal{R}_1$ terminiert, denn $l \succ_{\text{lpos}} r$ für alle $l \to r \in \mathcal{R}_0$ und auch für (G4).
Die kritischen Paare aus $CP(\mathcal{R}_0)$ sind mit $\mathcal{R}_1$ alle zusammenführbar. $\mathcal{R}_1$ ist trotzdem nicht konfluent, da sich durch die neue Regel (G4) neue kritische Paare ergeben:
z.B.:



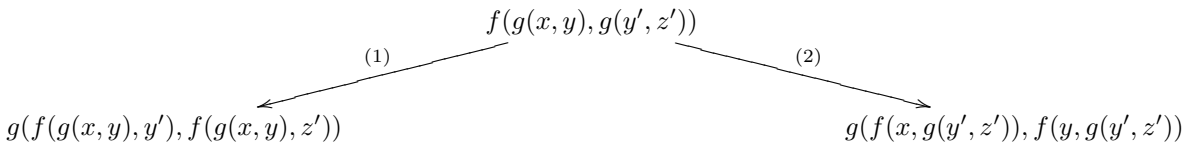
neues kritisches Paar, das nicht zusammenführbar ist. Insgesamt entstehen für $\mathcal{R}_1$ 5 kritische Paare, von denen 4 nicht zusammenführbar sind.
$\curvearrowright$ erzeuge daraus 4 neue Regeln (G5)-(G8) (eine davon ist $f(x, i(e)) \to x$).
$\mathcal{R}_2 = \{(G1) - (G8)\}$ etc.

3 Möglichkeiten für das Verhalten dieses Vervollständigungsalgorithmusses: Erfolg, Fehlschlag, Nicht-Terminierung
Erfolg: nach endlich vielen Schritten erreicht man ein konfluentes TES
Fehlschlag: Bei jeder neue erzeugten Regel $s' \to t'$ bzw $t' \to s'$ muss überprüft werden, ob das TES immer noch terminiert. Wenn die bisherigen Regeln mit der Reduktionsordnung $\succ$ terminierten, dann überprüfe, ob $s' \succ t'$ bzw $t' \succ s'$ ist. Abbruch, falls $s' \equiv t'$ mit der bisherigen Reduktionsordnung nicht orientiert werden kann. gegebenenfalls Neustart mit neuer Reduktionsordnung. Verbesserung: Lasse $\succ$ so unbestimmt wie möglich und lege sie nur dann weiter fest, wenn es zur Orientierung von $s' \equiv t'$ nötig ist.
Nicht-Terminierung: Scheitern wird "nicht bemerkt"
$\Rightarrow$ Algorithmus BASIC_COMPLETITION
Schritt (1) Richten der Gleichungen in $\mathcal{E}$
+2
Schritt (4) Vervollständigungsschritt Danach sind alle kritischen Paare von $\mathcal{R}_i$ zusammenführbar
Schritt (5) Erfolgsfall Alle kritischen Paare von $\mathcal{R}_i$ waren schon in $\mathcal{R}_i$ zusammenführbar $\curvearrowright \mathcal{R}_i$ konfluent
3 Beispiele für Erfolg / Fehlschlag / Nichtterminierung von BASIC_COMPLETITION

Beispiel 6.1.3: Erfolg
$\mathcal{E} = \{f(f(x,y), f(y,z)) \equiv y\}$ Zentrales Gruppoid
$\succ$ z.B. rpos Alle kritischen Paare von $\mathcal{R}_1$ sind zusammenführbar $\curvearrowright \mathcal{R}_2 = \mathcal{R}_1$. $\curvearrowright$ Erfolg: $\mathcal{R}_1$ ist konvergent und äquivalent zu $\mathcal{E}$.

Beispiel 6.1.4: Fehlschlag
$\mathcal{E} = \{f(x, g(y,z)) \equiv g(f(x,y), f(x,z)), f(g(x,y), z) = g(f(x,z), f(y,z))\}$ f ist links- und rechtsdistributiv über g
Wähle $\succ = \succ_{rpos}$ mit f]g

$\mathcal{E} = \{f(x, g(y, z)) \to g(f(x, y), f(x, z)), f(g(x, y), z) = g(f(x, z), f(y, z))\}$



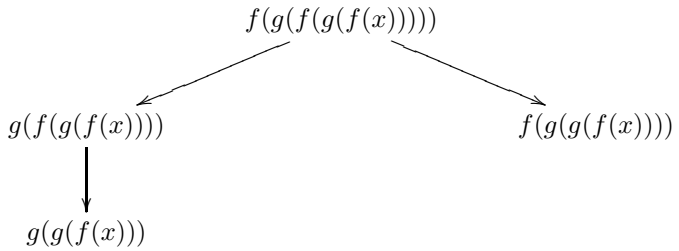
$g(g(f(x,y')f(y,y')), g(f(x,z'), f(y,z')))$ $\qquad$ $g(g(f(x,z'), f(x,z')), g(f(y,y'), f(y,z')))$

Hieraus lässt sich keine terminierende Regel gewinnen, da beide Terme gleich sind, wenn alle Terme gleich instantiiert werden $\Rightarrow$ Fehlschlag

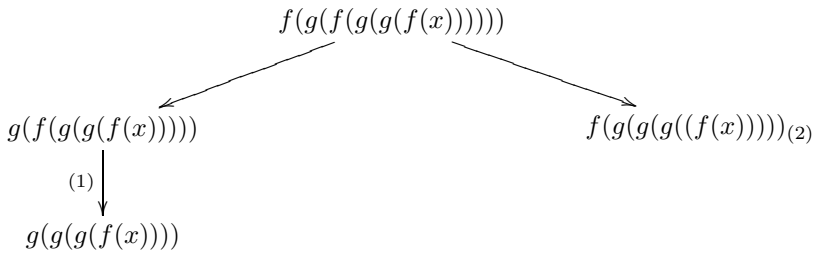
[20.07.04]

Vervollständigung: Finde ein konvergentes und zu $\mathcal{E}$ äquivalentes TES

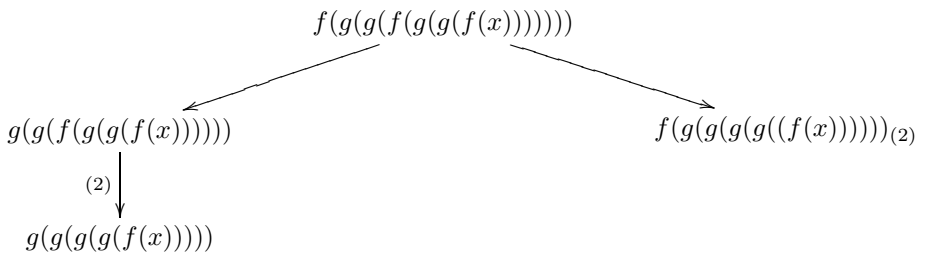
Beispiel 6.1.5: Nichtterminierung
$\mathcal{E} = \{f(g(f(x))) \equiv g(f(x))\}$ "Man darf das äussere f weglassen, wenn ein g zwischen zwei f's steht"
$\mathcal{R}_0 = \{f(g(f(x))) \to g(f(x))\}$ ($\succ$ beliebiige Simplifikatonsordnung)



$\mathcal{R}_1 = \{f(g(f(x))) \overset{(1)}{\to} g(f(x))f(g(g(f))))) \overset{(2)}{\to} g(g(f(x)))\}$ "Man darf das äussere f weglassen, wenn ein oder zwei g's zwischen zwei f's stehen"

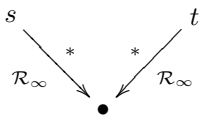


$f(g(g(f(g(f(x))))))$
führt zur selben neuen Regel

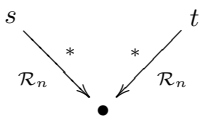


$\mathcal{R}_2 = \{f(g^i(f(x))) \to g^i(f(x)) | 1 \leq i \leq 4\}$ Vervollständigung kann unendlich oft wiederholt werden
$\curvearrowright$ erzeugt eine unendliche Menge von Regeln $\mathcal{R}_\infty = \{f(g^i(f(x))) \to g^i(f(x)) | 1 \leq i\}$

Selbst wenn der Vervollständigungsalgorithmus nicht terminiert, entsteht ein unendliches, aber konvergentes TES $\mathcal{R}_\infty$, das äquivalent zu $\mathcal{E}$ ist $\Rightarrow$ Man kann den Vervollständigungsalgorithmus dann immer noch als Semi-Entscheidungsverfahren für das Wortproblem benutzen: Um $s \equiv_\mathcal{E} t$ zu zeigen, kann man wie folgt vorgehen:
$s \equiv_\mathcal{E} t$ gdw.



gdw ein n existiert, so dass



Semi-Entscheidungsverfahren: Untersuche, ob $s \downarrow_{\mathcal{R}_0} t$. Wenn ja, gilt $s \equiv_\mathcal{E} t$. Wenn nein, untersuche $s \downarrow_{\mathcal{R}_1} t$ etc

Satz 6.1.6:
Sei $\mathcal{E}$ ein (endliches) Gleichungssystem, sei $\succ$ eine Reduktionsordnung.

a) Falls BASIC_COMPLETITION($\mathcal{E}$, $\succ$) terminiert und $\mathcal{R}_n$ als Resultat liefert, dann ist $\mathcal{R}_n$ ein (endliches) konvergentes TES, das äquivalent zu $\mathcal{E}$ ist.

b) Falls BASIC_COMPLETITION($\mathcal{E}$, $\succ$) nicht terminiert, dann ist $\mathcal{R}_\infty = \cup_{i \geq 0} \mathcal{R}_i$ ein unendliches konvergentes TES, das äquivalent zu $\mathcal{E}$ ist

(Fall (a): Wir erhalten ein Entscheidungsverfahren für das Wortproblem von $\mathcal{E}$
Fall (b): Wir erhalten ein Semi-Entscheidungsverfahren für das Wortproblem von $\mathcal{E}$)
Beweis: (Kurzfassung)

a) $\mathcal{R}_n$endlich: in jedem Schleifendurchlauf kommen nur endlich viele Regeln dazu

   $\mathcal{R}_n$ terminiert: $l > r$ gilt für alle Regeln

   $\mathcal{R}_n$ konfluent: alle kritischen Paare von $\mathcal{R}_n$ zusammenführbar (kritische Paar Lemma, Diamond Lemma)

   $\mathcal{R} - n$ äquivalent zu $\mathcal{E}$: alle kritischen Paare sind „korrekt“ bezüglich $\mathcal{E}$

b) $\mathcal{R}_\infty$ unendlich: in jedem Schleifendurchlauf kommen neue Regeln dazu

   $\mathcal{R}_\infty$ konfluent: jedes $\langle s, t\rangle \in CP(\mathcal{R}_\infty)$ ist kritisches Paar von einem $\mathcal{R}_n \curvearrowright$ in $\mathcal{R}_{n+1}$ zusammenführbar.

   $\mathcal{R}_\infty$ terminiert: wie (a)

   $\mathcal{R}_\infty$ äquivalent zu $\mathcal{E}$: wie (a)

□

Nachteil von BASIC_COMPLETION: Regeln, die man einmal gebildet hat (in einem $\mathcal{R}_i$) kann man nicht mehr verändern bzw. löschen $\curvearrowright$ man muss kritische Paare mit all diesen Regeln bilden.
→ zu aufwendig (und zu schwach) für realistischen Einsatz.

6.2 Ein vernesserter Vervollständigungsalgorithmus

Ziel ist: Wenn man neue Regeln erzeugt, dann sollte man alte Regeln mit Hilfe der neuen Regeln vereinfachen bzw. löschen.

Beispiel 6.2.1:
Gruppenbeispiel [Folie]

(G5) ein Spezialfall der Tatsache, dass$i(e) \equiv e$
(G6) ein Spezialfall der Tatsache, dass$i(e) \equiv e$
(G7) ist die Erkenntnis$i(e) \equiv e$

Man kann mit neuen Regeln (G7) alte Regeln vereinfachen

$$f(x, i(e)) \to x \quad (G5)$$
$$\downarrow (G7)$$
$$f(x, e)$$
$$\downarrow (G2)$$
$$x$$

Es entsteht: $x \equiv x$ Triviale Aussagen kann man löschen

Vervollständigungsalgorithmus: Sammlung von 6 Transformationsregeln auf Paaren $(\mathcal{E}, \mathcal{R})$. $\mathcal{E}$: endliches Gleichungssystem $\mathcal{R}$. endliches TES
Transformation beginnt mit $(\mathcal{E}, \emptyset)$ $\underbrace{\vdash_c}_{\text{Überführung durch Anwendern von Transformationsregeln}(\mathcal{E},\mathcal{R})\vdash_c(\mathcal{E}',\mathcal{R}')}$ $\ldots \vdash_c$
$(\emptyset, \mathcal{R})$.
Dann ist $\mathcal{R}$ das konvergente, zu $\mathcal{E}$ äquivalente TES (unter besonderen Bedingungen)
Bei $(\mathcal{E}, \mathcal{R}) \vdash_c (\mathcal{E}', \mathcal{R}')$ ändert sich die Menge der wahren Gleichungen nicht, d.h. $\leftrightarrow^*_{\mathcal{E}\cup\mathcal{R}} = \leftrightarrow^*_{\mathcal{E}'\cup\mathcal{R}'}$
Intuition: $\mathcal{E}$ enthält die Gleichungen, die noch nicht gerichtet wurden
$\mathcal{R}$ enthält die Gleichungen, die schon nicht gerichtet wurden (als Regeln)

Definition 6.2.2: Transformationsregeln für Vervollständiung
Sei $\mathcal{E}$ ein Gleichungssystem, $\mathcal{R}$ ein TES, $\succ$ eine Reduktionsordnung. Dann definieren wir folgende Transformationsregeln auf Paaren $(\mathcal{E}, \mathcal{R})$
[Folie]
(Die ersten 4 Transformationsregeln „entsprechen“ dem grundlegenden Vervollständigungsalgorithmus)
(Reduziere-Rechts ist unkritisch für Terminierung: Wenn man mit $(\mathcal{E}, \emptyset)$ startet, dann gilt $l \succ r$ für alle Regeln $l \to r \in \mathcal{R}$. D.h.: $s \succ t$, $t \succ v \curvearrowright s \succ v \Rightarrow$ Terminierung belibt erhalten.
Bei Reduziere-links: $u \succ t$ ist nicht sichergestellt $\curvearrowright$ anstelle der Regel $u \to t$ erzeugt man eine Gelichung $u \equiv t$, die dann noch orientiert werden muss.
Grund für die Einschränkung, dass l nicht mit $s \to t$ reduziert werden kann $\to$ später
Wir schreiben $(\mathcal{E}, \mathcal{R}) \vdash_c (\mathcal{E}', \mathcal{R}')$ falls $(\mathcal{E}, \mathcal{R})$ durch eine der Transformationsregeln in $(\mathcal{E}', \mathcal{R}')$ Überführt werden kann.

Beispiel 6.2.3:
$\mathcal{E} = \{h(x,y) \equiv f(x), h(x,y) \equiv f(y), g(x,y) \equiv h(x,y), g(x,y) \equiv a\}$.
$(\epsilon, \mathcal{R}) \underbrace{\vdash_c^*}_{\text{orientieren}} (\emptyset, \{h(x,y) \overset{(1)}{\to} f(x), h(x,y) \overset{(2)}{\to} f(y), g(x,y) \overset{(3)}{\to} h(x,y), g(x,y) \overset{(4)}{\to} a\})$
Generieren: erzeugt $f(x) \equiv f(y)$ aus (1) und (2) $(\{f(x) \underbrace{\equiv}_{\text{kann man nicht zu terminierender Regel orientieren}\curvearrowright\text{BASIC_}} a\}, \underbrace{\ldots}_{\text{wie oben}}$
Orientieren. überführe $h(x,y) \equiv a$ in $h(x,y) \to a$
Reduziere-rechts $(\{f(x) \equiv f(y), a \overset{(1)}{\equiv} f(x), a \overset{(2)}{\equiv} f(y)\}$
Reduziere-links $\{g(x,y) \to a, h(x,y) \to a\})$
Orientieren: $(\{f(x) \equiv f(y)\}, \{f(x)' a, q(x,y) \to a, a \equiv f(y), h(x,y) \to a\})$
Lösung: $\{f(x) \to a, q(x,y) \to a, h(x,y) \to a\}$.
Reduktionsgleichung und Löschen: $(\emptyset, \{f(x) \to a, q(x,y) \to a, h(x,y) \to a\})$

[27.07.04]

Verbersserte Vervollständigung:

- arbeitet auf Paaren $(\mathcal{E}, \mathcal{R})$
- Start mit $(\mathcal{E})$
- Erfolg mit $(\emptyset, \mathcal{R})$, $\mathcal{R}$ konvergent und äquivalent zu $\mathcal{E}$

Lemma 6.2.4:
Sei $\succ$ die bei der Vervollständigung benutzte Reduktionsordnung. Falls $l \succ r$ für alle $l \to r \in \mathcal{R}$ und $(\mathcal{E}, \mathcal{R}) \vdash_c (\mathcal{E}', \mathcal{R}')$, dann gilt $l' \succ r'$ für alle $l' \to r' \in \mathcal{R}'$.
Beweis: offensichtlich, bei „Reduziere-rechts“ wegen Transitivität von $\succ$
□

Lemma 6.2.5:
Aus $(\mathcal{E}, \mathcal{R}) \vdash_c (\mathcal{E}', \mathcal{R}')$ folgt $lras_{\mathcal{E} \cup \mathcal{R}} = lras_{\mathcal{E}' \cup \mathcal{R}'}$ (Bei $(\mathcal{E}, \emptyset) \vdash_c^* (\emptyset, \mathcal{R})$ gilt $\leftrightarrow_{\mathcal{E}}^* = \leftrightarrow_{\mathcal{R}}^*$, d.h. $\mathcal{R}$ ist äquivalent zu $\mathcal{E}$)
Beweis: einfach

Bei $(\mathcal{E}, \emptyset) \vdash_c^* (\emptyset, \mathcal{R})$ ist $\mathcal{R}$ nicht unbedingt konfluent:

$$\mathcal{E} = \{a \equiv b, a \equiv c\}\ (\mathcal{E}, \emptyset) \overset{\text{orientieren}}{\vdash_c^2} (\emptyset, \underbrace{\{a \leftarrow a \to c\}}_{\text{nicht konfluent}})$$

↷ Transformation führt nur dann zu konfluenten TES, wenn „alle“ kritischen Paare gebildet werden. Mann muss nur vor den Regeln kritische Paare bilden, die irgendwann gebildet werden, und nicht wieder gelöscht werden → „Persistente Regeln“. Definieren dies für endliche und unendliche Transformationsfolgen:

Definition 6.2.6: Persistente Gleichungen und Regeln

- Für eine endliche Folge $(\mathcal{E}_0, \mathcal{R}_0) \vdash_c (\mathcal{E}_1, \mathcal{R}_1) \vdash_c \ldots \vdash_c (\mathcal{E}_n, \mathcal{R}_n)$ definieren wir die persistenten Gleichungen $\mathcal{E}_\omega = \epsilon_n$ 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 $\mathcal{E}_\omega = \cup_{i \geq 0} \cap_{j \geq i} \mathcal{E}_j$ (die Gleichungen, die in $\mathcal{E}_j$ mit $j \geq i$ auftreten) und die persistenten Regeln $\mathcal{R}_\omega = \cup_{i \geq 0} \cap_{j \geq i} \mathcal{R}_j$.

Definition 6.2.7: Erfolg und Korrektheit von Vervollständigungsprozeduren

- Eine (endliche oder unendliche) Transformationsfolge ist <u>erfolgreich</u> $((\mathcal{E}_0, \mathcal{R}_0) \vdash_c (\mathcal{E}_1, \mathcal{R}_1) \vdash_c \ldots)$ gdw. $\mathcal{E}_\omega = \emptyset$ und $\mathcal{R}_\omega$ ein konvergentes zu $\mathcal{E}_0$äquivalentes TES ist.
- Eine Transformationsfolge

  Vervollständigungsprozeduren unterscheiden sich in der Strategie zur Anwendung der Transformationsregeln (→ beeinflusst Mächtigkeit und Effizienz).

  Aber: Korrektheit sollte garantiert sein.

Definition 6.2.8: Fairness

Eine (endliche oder unendliche) Transformationsfolge ist fair ($(\mathcal{E}_0, \mathcal{R}_0) \vdash_c (\mathcal{E}_1, \mathcal{R}_1) \vdash_c \ldots$) gdw. $CP(\mathcal{R}_\omega) \subseteq \cup_{i \geq 0} \mathcal{E}_i$. (d.h. jedes kritische Paar der persistenten Regeln trat mal als Gleichung auf)

Satz 6.2.9: Vervollständigungssatz

Jede Vervollständigungsprozedur, bei der jede nicht-fehlschlagende Transformationsfolge fair ist, ist korrekt.

Beispiel 6.2.10:

illustiert warum bei Reduziere-Links "$s \to t$"nur dann in "$u \equiv t$" überführt werden darf, wenn $s \overset{l \to r}{\to} u$ und l ist mit $s \to t$ nicht reduzierbar.

Durch kritische-Paar-Bildung entstehen die Regeln:

$fgf(x) \to fg(x)$ $\qquad$ ($fgf(x) \to fg^{2^n}(x)$ für alle $n \geq 0$)

$fgf(x) \to fg^2(x)$

$fgf(x) \to fg^4(x)$

$fgf(x) \to fg^8(x)$

$\vdots$

Zweite Regel kann linke Seite der ersten Regel reduzieren.

Erste Regel kann linke Seite der zweiten Regel reduzieren.

Ohne Einschränkung bei „Reduziere-links" könnte man die erste Regel löschen und die zweite behalten.

Später: Lösche dritte Regel mit der vierten, lösche vierte Regel mit der fünften, ...

$\Rightarrow$ einzige persistente Regel, keine persistenten Gleichungen: $g^2(x) \to g(x)$.

Transformationsfolge ist fair. Aber: Ergebnis (persistente Regeln) ist nicht äquivalent zum ursprünglichen Gleichungssystem.

6.3 Implizite Induktion

Ziel: Verwende den Vervollständingungsalgorithmus für automatische Induktionsbeweise.

Motiv: in der Programmverifkation ist man meist nicht daran interessiert, ob $\mathcal{E} \vDash s \equiv t$ gilt ($s \equiv t$ gilt in allen Modellen von $\mathcal{E}$), sondern man möchte nur Gültigkeit von $s \equiv t$ in einem speziellen Modell von $\mathcal{E}$ untersuchen.

Beispiel 6.3.1:

$\mathcal{E}$-plus-Gleichungen. Untersuche, ob $plus(x, s(y)) \equiv s(plus(x, y))$

Algebra $A = (\mathbb{N}, \alpha)$ mit $\alpha_0 = 0$, $\alpha_s(n) = n + 1$, $\alpha_{plus}(n, m) = n + m$: $A \vDash \mathcal{E}$, $A \vDash plus(x, s(y)) \equiv s(plus(x, y))$

Aber: $\mathcal{E} \nvDash plus(x, s(y)) \equiv s(plus(x, y))$, denn es existieren Modelle B von $\mathcal{E}$ mit $B \nvDash plus(x, s(y)) \equiv s(plus(x, y))$.

z.B. $B = (\mathbb{N} \cup \{\square, \lozenge\}, \alpha')$ mit $\alpha'_0 = 0$, $\alpha'_s(n) =$

$$\begin{cases} n+1, & \text{falls } n \in \mathbb{N} \\ \lozenge, & \text{falls n} \in \{\square, \lozenge\} \end{cases}$$

$\alpha'_{plus}(n, m) =$

$$\begin{cases} n+m & n, m \in \mathbb{N} \\ n & n \in \{\square, \lozenge\} \\ m & n = 0, m \in \{\square, \lozenge\} \\ \lozenge & n > 0, m \in \{\square, \lozenge\} \end{cases}$$

$B \vDash \mathcal{E}$ (Übung)

$B \nvDash plus(x, s(y)) \equiv s(plus(x, y))$

Bei Variablenbelegung $\mathcal{B}$ mit $\mathcal{B}(x) = 0$

$$\begin{aligned} I = \mathbb{N} \cup \{\square, \lozenge\}, \alpha', \mathcal{B}): \quad & I(plus(x, s(y))) = \alpha'_{plus}(\square, \alpha'_s(0)) = \square \\ & I(s(plus(x, y))) = \alpha'_s(\underbrace{\alpha'_{plus}(\square, 0)}_{\square}) = \lozenge \end{aligned}$$

Programmverifikation: Gilt eine Aussage $s \equiv t$ für alle Belegungen der Variablen mit Datenonjekten (z.B. mit natürlichen Zahlen)

Ziel: Untersuche, ob eine Aussage für alle Instatiierungen der Variablen mit Datenobjekten/Grundtermen.

Definition 6.3.2: Induktive Gültigkeit

Sei $\mathcal{E}$ ein Gleichungssystem über Σ, $\mathcal{V}$ und $s, t \in \mathcal{T}(\Sigma, \mathcal{V})$, Die Gleichung $s \equiv t$ ist induktiv gültig in $\mathcal{E}(\mathcal{E} \vDash_I s \equiv t)$ gdw. $\mathcal{E} \vDash s\sigma \equiv t\sigma$ für alle Substitutionen σ mit $\sigma(x) \in \mathcal{T}(\Sigma, \mathcal{V})$ für alle $x \in \mathcal{V}(s) \cup \mathcal{V}(t)$.

Beispiel 6.3.3:

$\mathcal{E}$-plus-Gleichungen. $\mathcal{E} \vDash_I plus(x, s(y)) \equiv s(plus(x, y))$

Beweis: Zeige für alle grundterme t_1, t_2 $\qquad \mathcal{E} \vDash plus(t_1, s(t_2)) \equiv s(plus(t_1, t_2))$

Leichter Beweis durch strukturelle Induktion über t_1.

"Explizite" Induktion: Induktionsrelation, Induktionshypothese, etc. werden explizit gebildet und angewendet.

(Automatisierung der expliziten Induktion: APV)

Alternative: implizite Induktion: keine direkte Bildung von Induktionshypothese und Induktionsrelation

[30.07.04]

Satz 6.3.4: Konsistenzbeweismethode

Sei $\mathcal{E}$ ein Gleichungssystem über $\Sigma, \mathcal{V}, s, t \in \mathcal{T}(\Sigma, \mathcal{V})$.

Dann gilt $\mathcal{E} \models_I s \equiv t$ gdw. für alle Grundterme $u, v \in \mathcal{T}(\Sigma)$ mit $\mathcal{E} \not\models u \equiv v$ auch $\mathcal{E} \cup \{s \equiv t\} \not\models u \equiv v$ gilt. (d.h. falls $\mathcal{E} \cup \{s \equiv t\} \models u \equiv v$ dann auch $\mathcal{E} \models u \equiv v$)

Beweis: "$\Rightarrow$"

$\mathcal{E} \models_I s \equiv t$

$\mathcal{E} \cup \{s \equiv t\} \models u \equiv v$

$\overset{\text{Birkhoff}}{\curvearrowright} u = u_0 \leftrightarrow_{\mathcal{E} \cup \{s \equiv t\}} u_1 \leftrightarrow_{\mathcal{E} \cup \{s \equiv t\}} \ldots \leftrightarrow_{\mathcal{E} \cup \{s \equiv t\}} u_n = v$

$\overset{\text{Ersetze alle Variablen der } u_i \text{ durch Grundterme, } \leftrightarrow_{\mathcal{E} \cup \{s \equiv t\}} \text{ stabil}}{\curvearrowright} u = u\delta = u_0\delta \leftrightarrow_{\mathcal{E} \cup \{s \equiv t\}}$
$u_1\delta \leftrightarrow_{\mathcal{E} \cup \{s \equiv t\}} \ldots \leftrightarrow_{\mathcal{E} \cup \{s \equiv t\}} u_n\delta = v\delta = v$

$\overset{\text{da alle } u_i\delta \text{ Grundterme sind}}{\curvearrowright} u = u\delta = u_0\delta \leftrightarrow_{\mathcal{E} \cup \{s\sigma \equiv t\sigma | x\sigma \in \mathcal{T}(\Sigma \text{ für alle } x \in \mathcal{V}(s) \cup \mathcal{V}(t)\}}$
$u_1\delta \leftrightarrow_{\mathcal{E} \cup \{s\sigma \equiv t\sigma | x\sigma \in \mathcal{T}(\Sigma \text{ für alle } x \in \mathcal{V}(s) \cup \mathcal{V}(t)\}} u_2\delta \leftrightarrow_{\mathcal{E} \cup \{s\sigma \equiv t\sigma | x\sigma \in \mathcal{T}(\Sigma \text{ für alle } x \in \mathcal{V}(s) \cup \mathcal{V}(t)\}}$
$\ldots \leftrightarrow_{\mathcal{E} \cup \{s\sigma \equiv t\sigma | x\sigma \in \mathcal{T}(\Sigma \text{ für alle } x \in \mathcal{V}(s) \cup \mathcal{V}(t)\}} u_n\delta = v$

$\overset{\text{Birkhoff}}{\curvearrowright} \mathcal{E} \cup \{s\sigma \equiv t\sigma | x\sigma \in \mathcal{T}(\Sigma \text{ für alle } x \in \mathcal{V}(s) \cup \mathcal{V}(t)\} \models u \equiv v$

$\overset{\mathcal{E} \models_I s \equiv t}{\curvearrowright} \mathcal{E} \models u \equiv v$

"$\Leftarrow$"

$\mathcal{E} \cup \{s \equiv t\} \models u \equiv v \curvearrowright \mathcal{E} \models u \equiv v \qquad \mathcal{E} \models_I s \equiv t$

Es gilt: $\mathcal{E} \cup \{s \equiv t\} \models s\sigma \equiv t\sigma$ für alle σ, insbesondere für σ, die Variablen aus s, t mit Grundtermen instatiieren

$\overset{\text{Var}}{\curvearrowright} \mathcal{E} \models s\sigma \equiv t\sigma$ für alle Grundsubstitutionen $\sigma \curvearrowright \mathcal{E} \models_I s \equiv t$

□

Um Satz 6.3.4 zu automatisieren: Generiere ein konvergentes, zu $\mathcal{E}$ äquivalentes TES $\mathcal{R}$. Dann muss man Satz 6.3.4 nicht mehr für alle Grundterme u, v überprüfen, sondern nur für Grundnormalformen von RR

$NF(\mathcal{R}) = \{q \downarrow_{\mathcal{R}} | q \in \mathcal{T}(\Sigma)\}$

Im plus-Beispiel: $NF(\mathcal{R}) = \{\mathcal{O}, s(\mathcal{O}), s^2(\mathcal{O}), \ldots\}$

Satz 6.3.5: Konsistentbeweismethode mit konvergenten TESen

Sei $\mathcal{E}$ ein Gleichungssystem über $\Sigma, \mathcal{V}$, sei $\mathcal{R}$ ein zu $\mathcal{E}$ äquivalentes konvergentes TES, $s, t \in \mathcal{T}(\Sigma, \mathcal{V})$. Dann gilt: $\mathcal{E} \models_I s \equiv t$ gdw. für alle $q_1, q_2 \in NF(\mathcal{R})$ mit $q_1 \neq q_2$ auch $\mathcal{E} \cup \{s \equiv t\} \not\models q_1 \equiv q_2$ gilt.

Beweis: "⇒"

Vorraussetzung: $\mathcal{E} \models_I s \equiv t$.

z.Z.: $\mathcal{E} \cup \{s \equiv t\} \not\models q_1 \equiv q_2$ falls $q_1, q_2 \in NF(\mathcal{R})$ und $q_1 \neq q_2$

$\mathcal{E} \cup \{s \equiv t\} \models q_1 \equiv q_2$ für $q_1, q_2 \in NF(\mathcal{R})$

$\overset{\text{Satz 6.3.4}}{\curvearrowright}$ $\mathcal{E} \models q_1 \equiv q_2$

$\overset{\mathcal{R}\text{konvergent und äquivalent zu } \mathcal{E}}{\curvearrowright}$ $q_1 \downarrow_{\mathcal{R}} q_2$ (q_1, q_2 sind mit $\mathcal{R}$ zusammenführbar)

$\overset{q_1, q_2 \text{ Normalformen}}{\curvearrowright}$ $q_1 = q_2$

"⇐"

Vorraussetzung: für alle $q_1 \neq q_2 \in NF(\mathcal{R})$ gilt $\mathcal{E} \cup \{s \equiv t\} \not\models q_1 \equiv q_2$.
z.Z. $\mathcal{E} \models_I s \equiv t$

Annahme: $\mathcal{E} \not\models_I s \equiv t$

$\overset{\text{Satz 6.3.4}}{\curvearrowright}$ es existieren Grundterme u, v mit $\mathcal{E} \cup \{s \equiv t\} \models u \equiv v$, $\mathcal{E} \not\models u \equiv v$

Zeige nun, dass es dann statt u, v auch entsprechende Grundnormalformen q_1, q_2 gibt mit

a) $q_1 \neq q_2$
b) $\mathcal{E} \cup \{s \equiv t\} \models q_1 \equiv q_2$

Wähle $q_1 = u \downarrow_{\mathcal{R}}$, $q_2 = v \downarrow_{\mathcal{R}}$ (existiert, denn $\mathcal{R}$ ist konvergent)

a) $q_1 \neq q_2$, sonst $u \downarrow_{\mathcal{R}} v \curvearrowright \mathcal{E} \models u \equiv v \rightarrow \lightning$ zu (1)
b)

$$\left.\begin{array}{l} \mathcal{E} \cup \{s \equiv t\} \models q_1 \equiv u \\ \mathcal{E} \cup \{s \equiv t\} \models q_1 \equiv u \end{array}\right\} \text{da } \mathcal{R} \text{ äquivalent zu } \mathcal{E} \text{ wegen (2): } \mathcal{E} \cup \{s \equiv t\} \models q_1 \equiv q_2$$

□

Problem: Wie erkennt man $NF(\mathcal{R})$?

Lösung: Einschränkung auf TES $\mathcal{R}$ bestimmter Gestalt.

Vorgehen: Teile Σ in zwei disjunkte Teilmengen auf:

Σ^c: Konstruktoren, dienen zum Aufbau von Datenstrukturen z.B. $\mathcal{O}$, succ
Konstruktorgrundterme $\widehat{=}$ Datenobjekte

Σ^d: definierte Funktionssymbole, entsprechen den Algorithmen, z.B. plus

Definition 6.3.6: Definitionsprinzip

Sei $\Sigma^c \subseteq \Sigma$ mit $\Sigma_0^c \neq \emptyset$ und $|\sigma^c| \geq 2$,

Sei $\Sigma^d = \Sigma \backslash \Sigma^c$. Ein TES $\mathcal{R}$ über $\Sigma, \mathcal{V}$ genügt dem Definitionsprinzip gdw.

a) für alle $l \to r \in \mathcal{R}$ gilt$l \notin \mathcal{T}(\Sigma^c, \mathcal{V})$ (l muss ein def. Funktionssymbol enthalten)

b) für alle $f \in \Sigma^d$ und alle $t_1, t_2 \in \mathcal{T}(\Sigma^c)$ existiert eine Regel $l \to r \in \mathcal{R}$, so dass l den Term $f(t_1, \ldots, t_n)$ matcht (d.h. f ist vollständig)

Beispiel 6.3.7:

plus-Beispiel erfüllt Definitions-prinzip für $\Sigma^c = \{\mathcal{O}, succ\}$, $\Sigma^d = \{plus\}$.

Satz 6.3.8: Grundnormalformen bei Definitions-prinzip

Sei $\mathcal{R}$ ein TES, das dem Definitionsprinzip genügt. Dann gilt $NF(\mathcal{R}) = \mathcal{T}(\Sigma^c)$

Beweis: $NF(\mathcal{R}) \subseteq \mathcal{T}(\Sigma^c)$:

Annahme: $q \in NF(\mathcal{R})$, q enthält definiertes Funktionssysmbol

Dann existiert $\pi \in Occ(q)$ mit $q|_\pi = f(t_1, \ldots, t_n)$ für $f \in \Sigma^d$, $t_1, \ldots, t_n \in \mathcal{T}(\Sigma^c)$

Dann ist $q|_\pi$ und damit q keine Normalform wegen Definition 6.3.6 (b)

$\mathcal{T}(\Sigma^c) \subseteq NF(\mathcal{R})$: folgt aus Definition 6.3.6 (a)

□

Ziel: überprüfe ob $\mathcal{E} \cup \{s \equiv t\} \vDash q_1 \equiv q_2$ für verschiedene Konstruktorgrundterme q_1, q_2 gilt. Falls ja: $\mathcal{E} \nvDash_I s \equiv t$ Falls nein: $\mathcal{E} \vDash_I s \equiv t$

Dazu benutze Vervollständigungsalgorithmus und modifiziere ihn so, dass er mit "False" abbricht, sobald Gleichungen entstehen, aus denen $q_1 \equiv q_2$ für verschiedene Konstruktorgrundterme folgt. Wenn er mit Erfolg terminiert, dann existiert ein zu $\mathcal{E} \cup \{s \equiv t\}$ äquivalentes, konvergentes TES, aus dem $q_1 \equiv q_2$ nicht folgt. Starte hierbei nicht mit $(\mathcal{E} \cup \{s \equiv t\}, \emptyset)$, sondern mit $(\{s \equiv t\}, \underbrace{\mathcal{R}}_{\text{Konvergentes, zu } \mathcal{E} \text{ äquivalentes TES, dass das Definitionsprinzip erfüllt}})$

Änderungen:

- Orientieren nur noch dann, wenn die entstehende neue Regel links aussen ein definiertes Funktionssysbol hat (⇒ Definitionsprinzip Teil (a) soll erhalten beleiben)
- Zwei neue Transformationsregeln um Inkonsistenzen zu erkennen

- Inkonsistenz: $c_1(\ldots) \equiv c_2(\ldots) \curvearrowright$ verschiedene Konstruktorgrundterme wären dann "gleich" bezüglich $\mathcal{E} \cup \{s \equiv t\}$
  $c(\ldots) \equiv x$ da es Konstruktoren $c' \neq c$ gibt, folgt daraus $c(\ldots) \equiv c'(\ldots)$.
- Injektivität: "Orientieren" behandelt nur noch Gleichungen, bei denen aussen ein definiertes Funktionssymbol steht
  Injektivität behandelt Fall, dass links und rechts der gleiche Konstruktor steht.

Beispiel 6.3.9:

Starte Induktionsbeweiskalkül mit $(\{plus(x, s(y)) \equiv s(plus(x, y))\}, \mathcal{R})$

Da Vervollständigung mit einem konvergenten TES endet und keine Inkonsistenz entdeckt wird: $\mathcal{E} \vDash_I p(\ldots) \equiv s(plus(x, y))$

Satz 6.3.10:

Sei $\mathcal{E}$ ein Gleichungssystem über $\Sigma, \mathcal{V}$, $\mathcal{R}$ ein zu $\mathcal{E}$ äquivalentes, konvergentes TES, dass das Definitionsprinzip erfüllt, $s, t \in \mathcal{T}(\Sigma, \mathcal{V})$, Reduktionsordnung $\succ$. Sei $(\{s \equiv t\}, \mathcal{R}) \vdash_I \ldots$ eine faire Transformationsfolgemit dem Induktionsbeweiskalkül. Falls die Ableitung mit "False" abbricht, gilt $\mathcal{E} \nvDash_I s \equiv t$. Terminiert die Ableitung mit $(\emptyset, \mathcal{R}_\omega)$, dann gilt $\mathcal{E} \vDash_I s \equiv t$.