Induktives Theorembeweisen

Vorlesung 2 Stunden
WS 2003/2004

J. Avenhaus
Universität Kaiserslautern

Inhaltsverzeichnis

1 Einleitung 1
2 Initiale Algebren zu Spezifikationen 6
2.1 Initiale Algebren 7
2.2 Hilfsmittel für Beweise in der induktiven Theorie . . . 26
3 Beweisverfahren für induktive Theoreme 38
3.1 Explizite Induktion 40
3.2 Die Coverset-Methode 50
3.3 Induktionsbeweise durch Konsistenzprüfung 70
4 Positiv/negativ bedingte Spezifikationen 87
4.1 Motivation 87
4.2 Syntax und Semantik von Spezifikationen 92
4.3 Der Ansatz von QUODLIBET 115
4.4 Die Inferenzregeln 132
4.5 Die Beweisführung 172
Literatur 189

1 Einleitung

Motivation

Sei sig $= (S, F, \alpha)$ eine Signatur.
E ein bedingtes Gleichungssystem über sig.
spec $=$ (sig, E) eine Spezifikation.

Frage

Was spezifiziert spec?

Welche sig-Algebren $\mathcal{A} \in$ Mod(spec) werden betrachtet?

Sei

$s = t$ eine Gleichung über sig.

Frage Was heißt „$s = t$ gilt in spec“?

Mögliche Präzisierungen zu „$s = t$ gilt in spec“

1. Präzisierung:

Gültigkeit in Mod(spec).

Für alle $\mathcal{A} \in$ Mod(spec) gilt $\mathcal{A} \models s = t$.

2. Präzisierung:

Gültigkeit in $\mathcal{L} \subseteq$ Mod(spec).

Für alle $\mathcal{A} \in \mathcal{L}$ gilt $\mathcal{A} \models s = t$.

Beispiel: $\mathcal{L} = \{\mathcal{A} \in$ Mod(spec) $\mid \mathcal{A}$ termerzeugt $\}$.

3. Präzisierung:

Gültigkeit in einer speziellen Algebra $\mathcal{A}$.

$\mathcal{A} \models s = t$.

1. Beispiel: $\mathcal{A} =$ Initiales Modell von spec.

2. Beispiel: $\mathcal{A} = (\mathbb{Z}, \mathbb{Z}^*)$.

Fragen

Wie hängen die Gültigkeitsbegriffe zusammen?

Wie zeigt man „$s = t$ gilt in spec“ zu den verschiedenen Präzisierungen?

Beispiel: Listen über $\mathbb{N}$

Sei spec $=$ (sig, E).
sig $= (S, F, \alpha)$ mit
$S = \{\mathsf{NAT}, \mathsf{LIST}\}$, $F = \{0, s, +, \mathsf{nil}, ., \mathsf{app}\}$.

$$E : 0 + y = y \qquad \mathsf{app}(\mathsf{nil}, l_2) = l_2$$
$$s(x) + y = s(x + y) \qquad \mathsf{app}(x.l_1, l_2) = x.\mathsf{app}(l_1, l_2)$$

$\mathcal{A} = \mathcal{I}$(spec) das initiale Modell zu spec.

Die Gleichungen

$$x + y = y + x$$
$$\mathsf{app}(\mathsf{app}(l_1, l_2), l_3) = \mathsf{app}(l_1, \mathsf{app}(l_2, l_3))$$

- gelten in $\mathcal{A}$,
- gelten in allen termerzeugten Modellen von spec,
- gelten nicht in allen Modellen von spec,
- gelten in $\mathcal{A}' = (\mathbb{Z}, \mathbb{Z}^*)$ (bei „natürlicher“ Interpretation der $f \in F$).

Beispiel: Sortieren von Listen

Sei spec $=$ (sig, E).
sig $= (S, F, \alpha)$ mit
$S = \{\mathsf{BOOL}, \mathsf{NAT}, \mathsf{LIST}\}$,
$F = \{\mathsf{true}, \mathsf{false}, 0, s, \leq, \mathsf{nil}, ., \mathsf{ins}, \mathsf{isort}, \mathsf{sorted}\}$.

$$\begin{aligned} E :\ & 0 \leq y = \mathsf{true} \\ & s(x) \leq 0 = \mathsf{false} \\ & s(x) \leq s(y) = x \leq y \\ & \mathsf{ins}(x, \mathsf{nil}) = x.\mathsf{nil} \\ & \mathsf{ins}(x, y.l) = x.y.l \Leftarrow x \leq y = \mathsf{true} \\ & \mathsf{ins}(x, y.l) = y.\mathsf{ins}(x, l) \Leftarrow x \leq y = \mathsf{false} \\ & \mathsf{isort}(\mathsf{nil}) = \mathsf{nil} \\ & \mathsf{isort}(x.l) = \mathsf{ins}(x, \mathsf{isort}(l)) \end{aligned}$$

$$\mathsf{sorted}(\mathsf{nil}) = \mathsf{true} \qquad \mathsf{sorted}(x.y.l) = \mathsf{false} \Leftarrow x \leq y = \mathsf{false}$$
$$\mathsf{sorted}(x.\mathsf{nil}) = \mathsf{true} \qquad \mathsf{sorted}(x.y.l) = \mathsf{sorted}(y.l) \Leftarrow x \leq y = \mathsf{true}$$

$\mathcal{A} = \mathcal{I}$(spec) das initiale Modell von spec.

Dies spezifiziert das Sortierverfahren *insertion sort*.

Es gilt

$\mathcal{A} \models$ sorted (isort(l))=true.

Frage

Wie zeigt man dies automatisch?

Beispiel: Schlange ohne Wiederholungen von Einträgen

Sei spec $= (\text{sig}, E)$.
sig $= (S, F, \alpha)$ mit
$S = \{\text{BOOL}, \text{NAT}, \text{LIST}\}$,
$F = \{\text{true}, \text{false}, 0, s, \text{nil}, ., \text{top}, \text{pop}, \text{push}, \text{elem}\}$.

$$
\begin{aligned}
E :\ & \text{top}(x.l) = x \\
& \text{pop}(x.l) = l \\
& \text{push}(x, \text{nil}) = x.\text{nil} \\
& \text{push}(x, y.l) = y.l \Leftarrow x = y \\
& \text{push}(x, y.l) = y.\text{push}(x, l) \Leftarrow x \neq y \\
& \text{elem}(x, \text{nil}) = \text{false} \\
& \text{elem}(x, y.l) = \text{true} \Leftarrow x = y \\
& \text{elem}(x, y.l) = \text{elem}(x, l) \Leftarrow x \neq y
\end{aligned}
$$

$\mathcal{A} = \mathcal{I}(\text{spec})$.

Also

top und pop sind nur partiell spezifiziert.

spec benutzt negative Bedingungen, z. B. $\neg x = y$.

Frage

Gilt (1) oder (2)?

(1) $\mathcal{A} \models \text{push}(x, \text{push}(y, l)) = \text{push}(y, \text{push}(x, l))$.

(2) $\mathcal{A} \models \text{elem}(x, l) = \text{true} \Rightarrow \text{push}(x, l) = l$.

2 Initiale Algebren zu Spezifikationen

Ziel von Kapitel 2

„Initiale Algebra“ hat in der Modelltheorie eine feste Bedeutung. Wir definieren dies.
Ist spec eine positiv-bedingte Spezifikation, so ist $\mathcal{I}(\text{spec})$ initial in der Klasse Mod(spec).

Wir zeigen den Zusammenhang auf zwischen „gültig in $\mathcal{I}(\text{spec})$“ und „gültig in allen termerzeugten Modellen von spec“.

Sei R ein terminierendes und grund-konfluentes Regelsystem. Dann definiert R eine kanonische Termalgebra, die isomorph zu $\mathcal{I}(\text{spec})$ ist.

Wir entwickeln erste Hilfsmittel zum Beweis zu „$s = t$ ist gültig in $\mathcal{I}(\text{spec})$“.

Voraussetzung in Kapitel 2

spec ist eine positiv-bedingte Spezifikation.

2.1 Initiale Algebren

Erinnerung

Sei sig $= (S, F, \alpha)$ eine feste **Signatur** mit

S=Menge von Sortensymbolen.

F=Menge von Funktionssymbolen.

$\alpha : F \to S^+$ Angabe der Stelligkeit der $f \in F$.
$f : s_1, \ldots, s_n \to s$ steht für $\alpha(f) = s_1, \ldots, s_n, s$.

Eine **sig-Algebra** $\mathcal{A} = ((A_s)_{s \in S}, (f^{\mathcal{A}})_{f \in F})$
besteht aus

- Trägermengen A_s für die $s \in S$. Sei $A = \bigcup_{s \in S} A_s$.
- Funktionen $f^{\mathcal{A}}$ zu den $f \in F$ mit
$f^{\mathcal{A}} : A_{s_1} \times \cdots \times A_{s_n} \to A_s$, falls $\alpha(f) = s_1, \ldots, s_n, s$.

Also: $\mathcal{A}$ interpretiert

- die Sortensymbole $s \in S$ zu Mengen A_s,
- die Funktionssymbole $f \in F$ zu Funktionen $f^{\mathcal{A}}$.

sig-Homomorphismen (1)

Sei sig $= (S, F, \alpha)$ eine Signatur.
$\mathcal{A}, \mathcal{B}$ sig-Algebren.

Definition 2.1

Ein **sig-Homomorphismus** $\varphi : \mathcal{A} \to \mathcal{B}$ ist ein System $(\varphi_s)_{s \in S}$ von Funktionen $\varphi_s : A_s \to B_s$ mit

$\varphi_s(f^{\mathcal{A}}(a_1, \ldots, a_n)) = f^{\mathcal{B}}(\varphi_{s_1}(a_1), \ldots, \varphi_{s_n}(a_n))$,
falls $f : s_1, \ldots, s_n \to s$.

Wir schreiben $\varphi(a)$ statt $\varphi_s(a)$, falls $a \in A_s$.

φ ist ein sig-Homomorphismus von $\mathcal{A}$ **auf** $\mathcal{B}$, falls zu jedem $b \in B_s$ ein $a \in A_s$ existiert mit $\varphi(a) = b$.

φ ist ein **sig-Isomorphismus**, falls alle φ_s bijektiv sind.

Beispiel

Sei $\mathcal{A}$ die freie Termalgebra über sig, also $A_s = \mathrm{Term}_s(F)$ und $f^{\mathcal{A}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$. Sei φ gegeben durch

$$\varphi(f(t_1, \ldots, t_n)) = f^{\mathcal{B}}(\varphi(t_1), \ldots, \varphi(t_n))$$

Dann ist $\varphi : \mathcal{A} \to \mathcal{B}$ ein sig-Homomorphismus.

Schreibweise

$\varphi(t) = t^{\mathcal{B}}$ „Auswertung von $t \in \mathrm{Term}(F)$ in $\mathcal{B}$“.

sig-Homomorphismen (2)

Bezeichnung

Die sig-Algebra $\mathcal{A}$ **identifiziert** $t_1, t_2 \in \mathsf{Term}(F)$, falls $t_1^{\mathcal{A}} = t_2^{\mathcal{A}}$ gilt.

Lemma 2.1

Sei $\varphi : \mathcal{A} \longrightarrow \mathcal{B}$ ein sig-Homomorphismus:

a) Für alle $t \in \mathsf{Term}(F)$ gilt $\varphi(t^{\mathcal{A}}) = t^{\mathcal{B}}$.

b) Aus $t_1^{\mathcal{A}} = t_2^{\mathcal{A}}$ folgt $t_1^{\mathcal{B}} = t_2^{\mathcal{B}}$.

Beweis:

a) Induktion über den Aufbau von t.

$t \equiv c$: Es gilt $\varphi(c^{\mathcal{A}}) = c^{\mathcal{B}}$

$t \equiv f(t_1, \ldots, t_n)$: Es gilt

$$\varphi(f(t_1, \ldots, t_n)^{\mathcal{A}}) = \varphi(f^{\mathcal{A}}(t_1^{\mathcal{A}}, \ldots, t_n^{\mathcal{A}})) =$$
$$= f^{\mathcal{B}}(\varphi(t_1^{\mathcal{A}}), \ldots, \varphi(t_n^{\mathcal{A}})) =_{IV} f^{\mathcal{B}}(t_1^{\mathcal{B}}, \ldots, t_n^{\mathcal{B}}) =$$
$$= f(t_1, \ldots, t_n)^{\mathcal{B}}$$

b) Es gilt mit a)

$t_1^{\mathcal{B}} = \varphi(t_1^{\mathcal{A}}) = \varphi(t_2^{\mathcal{A}}) = t_2^{\mathcal{B}}$

Beispiel zu sig-Homomorphismen

Sei $\mathsf{sig} = (\{\mathsf{ANY}\}, \{0, s, +\}, \alpha)$.
$\mathcal{A} = (\mathbb{N}, \{0, +1, +\})$,
$\mathcal{B} = (\mathbb{Z}, \{0, +2, +\})$.

Es gilt

a) $\varphi : \mathcal{A} \longrightarrow \mathcal{B}$, $\varphi(n) = 2 \cdot n$ ist ein sig-Homomorphismus, denn

$\varphi(0^{\mathcal{A}}) = \varphi(0) = 0^{\mathcal{B}}$
$\varphi(s^{\mathcal{A}}(n)) = \varphi(n+1) = 2(n+1) = 2n+2 = s^{\mathcal{B}}(\varphi(n))$
$\varphi(n +^{\mathcal{A}} m) = 2(n+m) = 2n+2m = \varphi(n) +^{\mathcal{B}} \varphi(m)$

b) Es gibt keinen sig-Homomorphismus $\varphi : \mathcal{B} \longrightarrow \mathcal{A}$, denn sonst $\varphi : \mathcal{B} \rightarrow \mathcal{A}$.
$0 = \varphi(0) = \varphi(-2+2) = \varphi(-2) + \varphi(2)$ in $\mathcal{A}$, also $\varphi(2) = \varphi(-2) = 0$,
aber $\varphi(2) = \varphi(s^{\mathcal{B}}(0)) = s^{\mathcal{A}}(0) = 1 \neq 0$

sig-Homomorphismen (3)

Seien sig $= (S, F, \alpha)$.
$\mathcal{A}, \mathcal{B}$ sig-Algebren.

Definition 2.2

$\mathcal{A}$ heißt **termerzeugt**, wenn es zu jedem $a \in A$ ein $t_a \in \mathsf{Term}(F)$ gibt mit $a = t_a^{\mathcal{A}}$.

Beispiel wie auf letzter Folie.

$\mathcal{A}$ ist termerzeugt, $\mathcal{B}$ nicht.

Lemma 2.2

Sei $\mathcal{A}$ termerzeugt. Ist $\varphi : \mathcal{A} \longrightarrow \mathcal{B}$ ein sig-Homomorphismus, so ist φ gegeben durch $\varphi(t_a^{\mathcal{A}}) = t_a^{\mathcal{B}}$. Es gibt also in diesem Fall maximal einen sig-Homomorphismus.

Beweis:

Zu jedem $a \in A$ gibt es ein t_a mit $a = t_a^{\mathcal{A}}$. Nach Lemma 2.1 gilt $\varphi(t_a^{\mathcal{A}}) = t_a^{\mathcal{B}}$. Dadurch ist φ auf A eindeutig festgelegt.

Initiale Algebren

Sei $\mathcal{A}$ sig-Algebra.
$\mathcal{L}$ Klasse von sig-Algebren.

Definition 2.3

$\mathcal{A}$ heißt **initial** in $\mathcal{L}$, falls gilt

1. $\mathcal{A} \in \mathcal{L}$.
2. Zu jedem $\mathcal{B} \in \mathcal{L}$ gibt es genau einen sig-Homomorphismus $\varphi : \mathcal{A} \rightarrow \mathcal{B}$.

Lemma 2.3

Sind $\mathcal{A}$ und $\mathcal{B}$ initial in $\mathcal{L}$, so sind $\mathcal{A}$ und $\mathcal{B}$ sig-isomorph.

Beweis:

Die Identität $\mathsf{id}_{\mathcal{A}}$ ist der einzige sig-Homomorphismus $\varphi : \mathcal{A} \longrightarrow \mathcal{A}$. Es gibt sig-Homomorphismen $\varphi : \mathcal{A} \longrightarrow \mathcal{B}$ und $\psi : \mathcal{B} \longrightarrow \mathcal{A}$. Dann sind $\varphi \circ \psi : \mathcal{B} \longrightarrow \mathcal{B}$ und $\psi \circ \varphi : \mathcal{A} \longrightarrow \mathcal{A}$ sig-Homomorphismen.

Also gilt $\varphi \circ \psi = \mathsf{id}_{\mathcal{B}}$ und $\psi \circ \varphi = \mathsf{id}_{\mathcal{A}}$.

Also gilt $\psi = \varphi^{-1}$ und φ ist ein sig-Isomorphismus.

Bedingte Gleichheitsspezifikationen

Definition 2.4

Eine **(bedingte) Spezifikation** hat die Form spec $=$ (sig, E) mit

- sig $= (S, F, \tau)$ Signatur.
- E System bedingter Gleichungen.
- Eine bedingte Gleichung hat die Form
  $s = t \Leftarrow u_1 = v_1 \wedge \cdots \wedge u_n = v_n$

Schreibweise

Sei $u = (u_1, \ldots, u_n)$, $v = (v_1, \ldots, v_n)$:

- $s = t \Leftarrow u = v$ statt $s = t \Leftarrow u_1 = v_1 \wedge \cdots \wedge u_n = v_n$
- $s = t$ statt $s = t \Leftarrow \qquad (n = 0)$
- $\sigma(u) = \sigma(v)$ statt $\sigma(u_i) = \sigma(v_i)$, $i = 1, \ldots, n$, falls σ Substitution.

Ab jetzt: „Spezifikation“ meint „bedingte Spezifikation“.

Beachte

$s = t \Leftarrow u = v$ steht für $\forall^*(u = v \Rightarrow s = t)$

Sei $\mathcal{A}x_E = \{\forall^*(u = v \Rightarrow s = t) \mid s = t \Leftarrow u = v \text{ in } E\}$
spec $=$ (sig, E) steht für spec $=$ (sig, $\mathcal{A}x_E$)

Definition von $=_E$

Idee

- unbedingtes Gleichungssystem E_i definieren.
- E_i approximiert E bis zur Rekursionstiefe i.

Sei spec $=$ (sig, E)

Definition 2.5 der E_i und von $=_E$

- $E_0 = \emptyset$.
- $E_{i+1} = \{\sigma(s) = \sigma(t) \mid s = t \Leftarrow u = v \text{ in } E,\ \sigma(u) =_{E_i} \sigma(v)\}$
- $=_i \ = \ =_{E_i}$
- $s =_E t \text{ gdw } \exists i : s =_i t$

Beachte

a) Es gilt $E_i \subseteq E_{i+1}$. (Induktion nach i)

b) Die E_i sind **unbedingte** Gleichungssysteme, also sind die $=_i$ wohl-definiert (wie früher). Damit ist auch $=_E$ wohl-definiert.

Satz 2.1.

Es gilt stets

$$s =_E t \text{ gdw } \mathcal{A} \models s = t \text{ für alle } \mathcal{A} \in \text{Mod(spec)}$$

Beweis: Übung.

Initiale Modelle

Sei spec $=$ (sig, E) positiv-bedingte Spezifikation.
$\mathcal{L} =$ Mod(spec).

Beachte

Es gibt nach Lemma 2.3 (bis auf Isomorphie) höchstens eine initiale Algebra in $\mathcal{L}$. Wir zeigen, dass es eine gibt.

Definition 2.6

$\mathcal{I}$(spec) ist folgende sig-Algebra $\mathcal{I} = ((I_s)_{s\in S}, (f^{\mathcal{I}})_{f\in F})$

$I_s = \{[t] \mid t \in \mathsf{Term}_s(F)\}$ mit
$[t] = \{t' \in \mathsf{Term}(F) \mid t =_E t'\}$

$f^{\mathcal{I}}([t_1], \ldots, [t_n]) = [f(t_1, \ldots, t_n)]$.

Lemma 2.4

$\mathcal{I}$(spec) ist ein termerzeugtes Modell zu spec. Es gilt $t^{\mathcal{I}} = [t]$ für alle $t \in \mathsf{Term}(F)$.

Beweis: Übung.

Bemerkung zu $\mathcal{I}$(spec)

Bemerkung

Sei sig $= (S, F, \alpha)$, spec $=$ (sig, E).

$\mathcal{I}$(spec) hängt von sig und E ab, nicht nur von E.

Beispiel

Sei $S = \{\mathsf{ANY}\}$, $F = \{0, +\}$, $F' = \{0, 1, +\}$.

$E: 0+x = x \qquad x+y = y+x \qquad x+(y+z) = (x+y)+z$.

a) sig $= (S, F, \alpha)$, spec $=$ (sig, E).
Es besteht $\mathcal{I}$(spec) nur aus einem Element: [0].

b) sig$' = (S, F', \alpha)$ spec$' =$ (sig$'$, E).
Es besteht $\mathcal{I}$(spec) aus unendlich vielen Elementen:
$I_{\mathsf{ANY}} = \{[\underline{n}] \mid n \in \mathbb{N}\}$ mit $\underline{0} = 0$ und $\underline{n} = 1 + (1 + \cdots + (1+1)\cdots)$ für $n > 0$.

Das initiale Modell zu spec

Sei spec $=$ (sig, E) eine bedingte Spezifikation.

Satz 2.2.

$\mathcal{I} = \mathcal{I}(\text{spec})$ ist initial in Mod(spec).

Beweis:

Es gilt $\mathcal{I} \in$ Mod(spec) und $\mathcal{I}$ ist termerzeugt nach Lemma 2.4.

Nach Lemma 2.2 reicht also zu zeigen:
Ist $\mathcal{B} \in$ Mod(spec) und $\varphi : \mathcal{I} \longrightarrow \mathcal{B}$ definiert durch $\varphi(t^{\mathcal{I}}) = t^{\mathcal{B}}$, so ist φ ein sig-Homomorphismus.

Beachte: $t^{\mathcal{I}} = [t]$.

1. φ ist wohl-definiert: Sei $[s] = [t]$. Dann gilt $s =_E t$ und somit $s^{\mathcal{B}} = t^{\mathcal{B}}$, da $B \in$ Mod(spec).
2. φ ist ein sig-Homomorphismus: $\varphi(f^{\mathcal{I}}([t_1], \dots, [t_n])) = \varphi([f(t_1, \dots, t_n)]) = f(t_1, \dots, t_n)^{\mathcal{B}} = f^{\mathcal{B}}(t_1^{\mathcal{B}}, \dots, t_n^{\mathcal{B}}) = f^{\mathcal{B}}(\varphi([t_1]), \dots, \varphi([t_n]))$

Folgerung

Sei $\mathcal{I} = \mathcal{I}(\text{spec})$, $\mathcal{A} \in$ Mod(spec), $t_1, t_2 \in \text{Term}(F)$.

Aus $\mathcal{I} \models t_1 = t_2$ folgt $\mathcal{A} \models t_1 = t_2$.

Beweis:

Aus $\mathcal{I} \models t_1 = t_2$ folgt $[t_1] = [t_2]$, also $t_1^{\mathcal{I}} = t_2^{\mathcal{I}}$.
Dies liefert $t_1^{\mathcal{A}} = \varphi(t_1^{\mathcal{I}}) = \varphi(t_2^{\mathcal{I}}) = t_2^{\mathcal{A}}$, also $\mathcal{A} \models t_1 = t_2$.

Kanonische Termalgebren (1)

Beachte

Die Elemente $[t]$ von $\mathcal{I}(\text{spec})$ sind (i. Allg. unendliche) Mengen.

Es rechnet sich leichter mit Repräsentanten $\hat{t} \in [t]$ statt mit den $[t]$.

Frage: Gibt es natürliche (kanonische) Repräsentanten?

Sei spec $=$ (sig, R).
R terminierend und grundkonfluent.

Definition 2.7

Die **kanonische Termalgebra** $\mathcal{T} = \mathcal{T}(\text{spec}) = ((T_s)_{s \in S}, (f^{\mathcal{T}})_{f \in F})$ ist gegeben durch

$T_s = \{t \in \text{Term}_s(F) \mid t\ R\text{-irreduzibel}\}$.

$f^{\mathcal{T}}(t_1, \dots, t_n) = f(t_1, \dots, t_n) \downarrow$.

Kanonische Termalgebren (2)

Beispiel

$S = \{\text{NAT}\}$, $F = \{0, s, +, *\}$.

$$R: \begin{array}{ll} x + 0 \to x & x * 0 \to 0 \\ x + s(y) \to s(x + y) & x * s(y) \to (x * y) + x \end{array}$$

Es gilt für alle $t \in \text{Term}(F)$:

t ist irreduzibel gdw $t \equiv s^i(0)$ für ein $i \in \mathbb{N}$.

Also:

$T_{\text{ANY}} = \{s^i(0) \mid i \in \mathbb{N}\}$.

$$\begin{array}{ll} 0^{\mathcal{T}} = 0 & s^i(0) +^{\mathcal{T}} s^j(0) = s^{i+j}(0) \\ s^{\mathcal{T}}(s^i(0)) = s^{i+1}(0) & s^i(0) *^{\mathcal{T}} s^j(0) = s^{i*j}(0) \end{array}$$

Satz 2.3.

Sei R terminierend und grundkonfluent. Dann sind $\mathcal{I}(\text{spec})$ und $\mathcal{T}(\text{spec})$ isomorph.

Beweis:

$\varphi : \mathcal{I}(\text{spec}) \to \mathcal{T}(\text{spec})$ mit $\varphi([t]) = t\downarrow_R$ ist ein sig-Isomorphismus (Übung!)

Beispiel zu $\mathcal{T}(\text{spec})$

Sei spec $= (\text{sig}, R)$.
sig $= (\{\text{ANY}\}, \{0, s, +, *\}, \alpha)$.

$$R: \begin{array}{ll} x + 0 \to x & x * 0 \to 0 \\ x + s(y) \to s(x + y) & x * s(y) \to (x * y) + x \end{array}$$

Es ist R konvergent.

a) $\mathcal{A} = (\mathbb{N}, \{0, +1, +, \cdot\})$.
$\mathcal{A}$ ist isomorph zu $\mathcal{T}(\text{spec})$, also initial in Mod(spec).
Denn $\varphi : \mathcal{A} \to \mathcal{T}(\text{spec})$, $\varphi(n) = s^n(0)$ ist sig-Isomorphismus.

b) $\mathcal{B} = (\mathbb{Z}, \{0, +1, +, \cdot\})$.
Es gilt $\mathcal{B} \in \text{Mod(spec)}$, aber $\mathcal{B}$ ist nicht initial in Mod(spec).
Denn: Es gibt keinen sig-Homomorphismus $\varphi : \mathcal{B} \to \mathcal{A}$.

Beispiel zu $\mathcal{T}(\text{spec})$ (Forts.)

c) $\mathcal{B} = (\{0, 1, 2\}, \{0^{\mathcal{B}}, s^{\mathcal{B}}, +^{\mathcal{B}}, *^{\mathcal{B}}\})$
mit

$$
\begin{array}{ll}
0^{\mathcal{B}} = 0 & n +^{\mathcal{B}} m = n + m \bmod 3 \\
s^{\mathcal{B}}(n) = n + 1 \bmod 3 & n *^{\mathcal{B}} m = n \cdot m \bmod 3
\end{array}
$$

Es gilt $\mathcal{B} \in$ Mod(spec), aber $\mathcal{B}$ ist nicht initial in Mod(spec).

Denn: Es gibt keinen sig-Homomorphismus $\varphi : \mathcal{B} \to \mathcal{A}$.

Sonst gilt in $\mathcal{A}$ (also in $\mathbb{N}$):

$0 = \varphi(0^{\mathcal{B}}) = \varphi(s^3(0)^{\mathcal{B}}) = s^{\mathcal{A}}(\varphi(s^2(0)^{\mathcal{B}}) = 2 + 1 = 3$

$\lightning$

Standardmodelle

Bezeichnung

$\mathcal{A} \in$ Mod(spec) heißt ein **Standardmodell** zu spec, falls $\mathcal{A}$ termerzeugt ist.

Frage:

Sei $t_1, t_2 \in$ Term(F, V).

Wann gilt $t_1 = t_2$ in allen Standardmodellen von spec?

Beachte

a) $\mathcal{A} \models t_1 = t_2$ steht für $\mathcal{A} \models \forall x_1, \ldots, x_n : t_1 = t_2$.

b) Für Grundterme t_1, t_2 gilt: Aus $\mathcal{I}(\text{spec}) \models t_1 = t_2$ folgt $\mathcal{A} \models t_1 = t_2$ für alle $\mathcal{A} \in$ Mod(spec).

c) Vermutung: Gilt $\mathcal{I}(\text{spec}) \models \sigma(t_1) = \sigma(t_2)$ für alle $\sigma \in$ GSub$(t_1 = t_2)$, so gilt $\mathcal{A} \models t_1 = t_2$ für alle Standardmodelle $\mathcal{A}$.

Bezeichnung

GSub$(t_1 = t_2) = \{\sigma \mid \sigma(x) \in$ Term(F) für alle $x \in$ Var$(t_1, t_2)\}$.

Gültigkeit in allen Standardmodellen (1)

Sei $\text{spec} = (\text{sig}, E), \text{sig} = (S, F, \alpha)$.

Erinnerung

Eine Evaluierungsfunktion zur sig-Algebra $\mathcal{A}$ ist $\text{eval} : V \longrightarrow A$, fortgesetzt zu $\text{eval} : \text{Term}(F, V) \rightarrow A$.

$\mathcal{A} \models t_1 = t_2$ gdw $\text{eval}(t_1) = \text{eval}(t_2)$ in $\mathcal{A}$ für alle eval.

Lemma 2.5

Sei $\mathcal{I} = \mathcal{I}(\text{spec})$.

a) Zu jedem $\text{eval} : V \longrightarrow I$ gibt es eine Grundsubstitution σ_{eval} mit $\text{eval}(t) = [\sigma_{\text{eval}}(t)]$ für alle $t \in \text{Term}(F, V)$.
b) Zu jeder Grundsubstitution σ gibt es ein $\text{eval}_\sigma : V \longrightarrow I$ mit $\text{eval}_\sigma(t) = [\sigma(t)]$ für alle $t \in \text{Term}(F, V)$.

Beweis-Idee:

a) Sei $\text{eval}(x) = [t_x], t_x \in \text{Term}(F)$. Setze $\sigma(x) = t_x$.
b) Setze $\text{eval}_\sigma(x) = [\sigma(x)]$.

Induktion über den Aufbau von t liefert die Behauptung.

Gültigkeit in allen Standardmodellen (2)

Satz 2.4.

Für alle $s, t \in \text{Term}(F, V)$ sind äquivalent.

1. $s = t$ ist gültig in $\mathcal{I} = \mathcal{I}(\text{spec})$.
2. $\sigma(s) =_E \sigma(t)$ für alle $\sigma \in \text{GSub}(s = t)$.
3. $s = t$ ist gültig in allen Standardmodellen $\mathcal{A}$ von spec.

Beweis:

1. $\rightsquigarrow$ 2.:

Sei $\sigma \in \text{GSub}(s = t)$. Nach Lemma 2.5 gibt es eval_σ mit $[\sigma(s)] = \text{eval}_\sigma(s)$ und $\text{eval}_\sigma(t) = [\sigma(t)]$. Wegen $\mathcal{I} \models s = t$ gilt $\text{eval}_\sigma(s) = \text{eval}_\sigma(t)$, also $[\sigma(s)] = [\sigma(t)]$, also $\sigma(s) =_E \sigma(t)$.

3. $\rightsquigarrow$ 1.:

Dies ist trivial, da $\mathcal{I}$ ein Standardmodell ist.

Gültigkeit in allen Standardmodellen (3)

2. $\rightsquigarrow$ 3.:

Sei $\mathcal{A}$ Standardmodell und eval : $V \to A$.

Zu zeigen ist eval(s) = eval(t) in $\mathcal{A}$.

Es gibt $t_x \in$ Term(F) mit eval$(x) = t_x^{\mathcal{A}}$. Sei $\sigma \in$ GSub$(s = t)$ gegeben durch $\sigma(x) = t_x$.

a) Es gilt stets eval$(t_0) = \sigma(t_0)^{\mathcal{A}}$ (Induktion über t_0).

b) Aus $t_1^{\mathcal{I}} = t_2^{\mathcal{I}}$ und $t_1, t_2 \in$ Term(F) folgt stets $t_1^{\mathcal{A}} = t_2^{\mathcal{A}}$ (nach Lemma 2.1).

c) Aus $\sigma(s) =_E \sigma(t)$ folgt $[\sigma(s)] = [\sigma(t)]$,
also $\sigma(s)^{\mathcal{I}} = \sigma(t)^{\mathcal{I}}$, also $\sigma(s)^{\mathcal{A}} = \sigma(t)^{\mathcal{A}}$ nach b).

d) Nach Voraussetzung gilt $\sigma(s)^{\mathcal{A}} = \sigma(t)^{\mathcal{A}}$ wegen c),
also folgt eval$(s) = \sigma(s)^{\mathcal{A}} = \sigma(t)^{\mathcal{A}} =$ eval(t).

Also (Ergebnis über Standardmodellen)

Die Gültigkeit von $s = t$ in $\mathcal{I}$(spec) impliziert die Gültigkeit von $s = t$ in allen Standardmodellen.

2.2 Hilfsmittel für Beweise in der induktiven Theorie

Problem

Gegeben: spec = (sig, E), $s = t$.

Frage: Gilt $\mathcal{I}$(spec) $\models s = t$?

Definition 2.8

ITh(spec) $= \{s = t \mid \mathcal{I}$(spec) $\models s = t\}$ heißt die **induktive Theorie** zu spec.

Beachte

Sei $\mathcal{I} = \mathcal{I}$(spec).

a) Es gilt $\mathcal{I} \models s = t$ gdw $\sigma(s) =_E \sigma(t)$ für alle $\sigma \in$ GSub$(s = t)$. Dies ist i. Allg. ein unendlicher Test.

b) Induktion hilft häufig, den Test endlich zu machen. Daher der Name „induktive Theorie“.

Hier

Hilfsmittel entwickeln, um Beweise einfacher zu gestalten.

spec-Konstruktoren (1)

Sei $\text{spec} = (\text{sig}, E), \text{sig} = (S, F, \alpha)$.

Beachte

Es gibt verschiedene Definitionen von Konstruktor.

1. Ansatz: Man zeichnet $C \subseteq F$ aus, nennt die $f \in C$ Konstruktoren und definiert auf der „Datenmenge" $\text{Term}(C)$ Funktionen $g \in F - C$.

2. Ansatz: Man wählt $C \subseteq F$ möglichst klein, so dass jeder F-Grundterm E-gleich zu einem C-Grundterm ist.

Also

Sind alle $g \in F - C$ total definiert, so stimmen beide Ansätze überein.

Hier

Wir wählen den 2. Ansatz.

Definition 2.9

$C \subseteq F$ heißt eine **Konstruktormenge** zu $\text{spec} = (\text{sig}, E)$, falls zu jedem $t \in \text{Term}(F)$ ein $t' \in \text{Term}(C)$ existiert mit $t =_E t'$.
C heißt **frei**, falls für alle $t_1, t_2 \in \text{Term}(C)$ gilt: $t_1 =_E t_2$ gdw $t_1 \equiv t_2$.

spec-Konstruktoren (2)

Frage

Wie findet man eine sinnvolle Konstruktormenge?

Lemma 2.6

Sei $\text{spec} = (\text{sig}, R)$ terminierend. Gilt $t \in \text{Term}(C)$ für jeden irreduziblen Term $t \in \text{Term}(F)$, so ist C eine Konstruktormenge.

Beweis: klar.

Also

Sei R terminierend und grundkonfluent und sei
$\text{Irr}(R) = \{t \in \text{Term}(F) \mid t \text{ irreduzibel}\}$. Sei $C \subseteq F$ minimal mit $\text{Irr}(R) \subseteq \text{Term}(C)$. Dann ist C eine minimale Konstruktormenge.
Für $t_1, t_2 \in \text{Irr}(R)$ gilt $t_1 =_R t_2$ gdw $t_1 \equiv t_2$.

Gilt also $\text{Irr}(R) = \text{Term}(C)$, so ist C eine Menge von freien Konstruktoren.

Beispiele

1. Beispiel

$F = \{0, s, +, -\}$.

R: (1) $x + 0 \to x$ (3) $x - 0 \to x$
(2) $x + s(y) \to s(x + y)$ (4) $0 - y \to 0$
(5) $s(x) - s(y) \to x - y$

Sei spec $=$ (sig, R). Dann ist R konvergent und $C = \{0, s\}$ ist Menge von freien Konstruktoren.

Sei R' das System R ohne (4) und spec$'$ $=$ (sig, R'). Dann ist $t \equiv (0 - s(0)) + (0 - s(0))$ irreduzibel, also F die kleinste Konstruktormenge. Sie ist nicht frei.

2. Beispiel

$F = \{0, s, \mathsf{nil}, ., \mathsf{app}, \mathsf{rev}\}$.

R: $\mathsf{app}(\mathsf{nil}, l_2) \to l_2$ $\mathsf{app}(x.l_1, l_2) \to x.\mathsf{app}(l_1, l_2)$
$\mathsf{rev}(\mathsf{nil}) \to \mathsf{nil}$ $\mathsf{rev}(x.l) \to \mathsf{app}(\mathsf{rev}(l), x.\mathsf{nil})$

Sei spec $=$ (sig, R). Dann ist R konvergent und $C = \{0, s, \mathsf{nil}, .\}$ eine Menge von freien Konstruktoren.

Verwendung von Konstruktormengen

Sei spec $=$ (sig, E), sig $= (S, F, \alpha)$ und $C \subseteq F$.

Definition 2.10

σ heißt eine **C-Substitution**, falls $\sigma(x) \in \mathsf{Term}(C, V)$ für alle $x \in V$.

Sei C-GSub$(s = t) = \{\sigma \mid \sigma(x) \in \mathsf{Term}(C)$ für alle $x \in \mathsf{Var}(s, t)\}$.

Satz 2.5.

Sei $\mathcal{I} = \mathcal{I}$(spec), $s, t \in \mathsf{Term}(F, V)$ und C eine Konstruktormenge. Es gilt $\mathcal{I} \models s = t$ gdw $\sigma(s) =_E \sigma(t)$ für alle $\sigma \in C$-GSub$(s = t)$.

Beweis:

„$\curvearrowright$“ klar, wegen Satz 2.4.

„$\curvearrowleft$“ Sei $\tau \in$ GSub$(s = t)$. Es gibt $\sigma \in C$-GSub$(s = t)$ mit $\sigma(x) =_E \tau(x)$ für alle $x \in \mathsf{Var}(s, t)$, da C Konstruktormenge.

Also $\tau(s) =_E \sigma(s) =_E \sigma(t) =_E \tau(t)$.

Also $\mathcal{I} \models s = t$, wegen Satz 2.4.

Berechnung einer Konstruktormenge

Sei spec $= (\text{sig}, R)$, sig $= (S, F, \alpha)$.

Ziel wegen Satz 2.5:

Berechnung einer **kleinen** Konstruktormenge.

Lemma 2.7

Berechne $C_0, C_1, \dots \subseteq F$ so

$C_0 = \{c \mid c \in F \text{ ist irreduzible Konstante}\}$.

$$C_{i+1} = C_i \cup \{f \in F - C_i \mid \text{es gibt } t_1, \dots, t_n \in \text{Term}(C_i), \text{ so dass } f(t_1, \dots, t_n) \text{ irreduzibel}\}.$$

Es gibt $k \in \mathbb{N}$ mit $C_k = C_{k+1}$. Sei $C = C_k$, dann ist $C = C_i$ für alle $i \geq k$. Ist R terminierend, so ist C eine Konstruktormenge zu spec.

Beweis:

Wegen $C_0 \subseteq C_1 \subseteq \dots \subseteq F$ und F endlich gibt es solch ein k. Es gilt $t \in \text{Term}(C)$, falls $t \in \text{Term}(F)$ irreduzibel ist (Induktion nach $>_{TT}$).

Also: Zu jedem $t \in \text{Term}(F)$ gibt es $t' \in \text{Term}(C)$ mit $t \xrightarrow{*} t'$.

Also: C ist Konstruktormenge, falls R terminierend ist.

Bemerkung/Beispiel

Bezeichnung

$C = C_R$ aus Lemma 2.7 heißt die durch R definierte Konstruktormenge.

Beispiel

$F = \{0, s, +, -\}$.

R: (1) $x + 0 \to x$
(2) $x + s(y) \to s(x + y)$
(3) $x - 0 \to x$
(4) $0 - y \to 0$
(5) $s(x) - s(y) \to x - y$

a) Sei spec $= (\text{sig}, R)$. Es ergibt sich
$C_0 = \{0\}$ $\quad C_1 = \{0, s\} = C_2$ $\quad$ Also $C = \{0, s\}$.

b) R' sei R ohne (4). Sei spec$' = (\text{sig}, R')$.
Es ergibt sich
$C_0 = \{0\}$, $C_1 = \{0, s\}$, $C_2 = \{0, s, -\}$,
$C_3 = \{0, s, -, +\} = F$.
Also gilt $C = F$.

Beachte

R definiert $+$ und $-$ total (über $\text{Term}(C)$).

Beispiel für nicht freie Konstruktoren

Sei $S = \{\mathsf{INT}\}$, $F = \{0, s, p, +, -\}$.

R:
$$\begin{array}{ll} s(p(x)) \to x & x + 0 \to x \\ p(s(x)) \to x & x + s(y) \to s(x+y) \\ & x + p(y) \to p(x+y) \\ x - 0 \to x & \\ x - s(y) \to p(x-y) & \\ x - p(y) \to s(x-y) & \end{array}$$

spec $= (\mathsf{sig}, R)$.

Es gilt

$C = \{0, s, p\}$ ist die von R definierte Konstruktormenge.

C ist nicht frei, denn es gilt z. B. $p(s(0)) =_R s(p(0))$.

Verwendung von Lemmata

Sei spec $= (\mathsf{sig}, E)$.

Definition 2.11

$s = t \in ITh(\mathsf{spec})$ heißt ein **Lemma** zu spec.

Lemma 2.8

Gilt $L \subseteq ITh(\mathsf{spec})$ und $\mathsf{spec}' = (\mathsf{sig}, L \cup E)$, so gilt $ITh(\mathsf{spec}) = ITh(\mathsf{spec}')$.

Beweis:

„$\subseteq$": Sei $s = t \in ITh(\mathsf{spec})$. Dann gilt für alle $\sigma \in \mathsf{GSub}(s = t)$ $\sigma(s) =_E \sigma(t)$, also auch $\sigma(s) =_{E \cup L} \sigma(t)$.

Also gilt $s = t \in ITh(\mathsf{spec}')$.

„$\supseteq$": Sei $s = t \in ITh(\mathsf{spec}')$, $\sigma \in \mathsf{GSub}(s = t)$, also $\sigma(s) =_{E \cup L} \sigma(t)$. Jeder Beweisschritt $\tau(u) \vdash\!\!\dashv \tau(v)$ mit $u = v \in L$ in diesem Beweis kann ersetzt werden durch den Beweis zu $\tau(u) =_E \tau(v)$. Also gilt $\sigma(s) =_E \sigma(t)$.

Also: $s = t \in ITh(\mathsf{spec})$.

Zusammenfassung

Satz 2.6.

Sei spec $= (\text{sig}, E)$, sig $= (S, F, \alpha)$, $C \subseteq F$ eine Konstruktormenge und $L \subseteq ITh(\text{spec})$ eine Menge von Lemmata zu spec.

Gibt es zu jedem $\sigma \in C$-GSub$(s = t)$ einen Beweis zu $\sigma(s) =_{E \cup L} \sigma(t)$, so gilt $s = t \in ITh(\text{spec})$.

Beweis: Dies folgt aus Satz 2.5 und Lemma 2.8.

Beachte

Beide Prinzipien sind von entscheidender Wichtigkeit für das induktive Beweisen:

1. Verwendung von Konstruktoren: Dies schränkt die zu betrachtenden σ ein.
2. Verwendung von Lemmata: Dies hilft beim Beweisen von $\sigma(s) =_E \sigma(t)$.

Ein Beispiel

Sei spec $= (\text{sig}, R)$, $S = \{\text{NAT}\}$, $F = \{0, s, +\}$.

$R : x + 0 \to x \qquad x + s(y) \to s(x + y)$

Beh.: $x + y = y + x \in ITh(\text{spec})$.

Es gilt $C = \{0, s\}$ ist eine Konstruktormenge.

Beweis-Prinzip

Induktion über den Aufbau von $\sigma(x)$ bzw. $\sigma(y)$.

Lemmata

$L1 : 0 + y = y \qquad L2 : s(x) + y = s(x + y)$

Ein Beispiel (Forts.)

Beweis zu den Lemmata

Beweis zu $L1$: Induktion über $\sigma(y)$.

$\sigma(y) \equiv 0 :\quad 0 + 0 =_R 0$

$\sigma(y) \equiv s(t) :\quad 0 + s(t) =_R s(0 + t) =_{IV} s(t)$

Beweis zu $L2$: Induktion über $\sigma(y)$, sei $\sigma(x) = \bar{x}$.

$\sigma(y) \equiv 0 :\quad s(\bar{x}) + 0 =_R s(\bar{x}) =_R s(\bar{x} + 0)$

$$\begin{aligned} \sigma(y) \equiv s(t) :\quad s(\bar{x}) + s(t) &=_R s(s(\bar{x}) + t) =_{IV} s(s(\bar{x} + t)) \\ &=_R s(\bar{x} + s(t)) \end{aligned}$$

Beweis zur Beh.: Induktion über $\sigma(y)$, sei $\sigma(x) = \bar{x}$.

$\sigma(y) \equiv 0 :\quad \bar{x} + 0 =_R \bar{x} =_{L1} 0 + \bar{x}$

$$\begin{aligned} \sigma(y) \equiv s(t) :\quad \bar{x} + s(t) &=_R s(\bar{x} + t) =_{IV} s(t + \bar{x}) \\ &=_{L2} s(t) + \bar{x} \end{aligned}$$

3 Beweisverfahren für induktive Theoreme

Problemstellung

Gegeben: spec $=$ (sig, E), sig $= (S, F, \alpha)$
$C \subseteq F$ Konstruktormenge
$L \subseteq ITh$(spec) Lemmata
$s = t$

Frage: Gilt $s = t \in ITh$(spec)?

Ansatz

Zeige für alle $\sigma \in C$-GSub$(s = t)$, dass $\sigma(s) =_{E \cup L} \sigma(t)$ gilt.

Beweis-Idee

Sei $\{x_1, \ldots, x_n\} \subseteq$ Var(s, t).

Definiere Ordnung $\succ_{ind}$ auf Term$(C)^n$ und damit die Induktionsordnung $\succ_i$ durch
$\sigma \succ_i \tau$ gdw $(\sigma(x_1), \ldots, \sigma(x_n)) \succ_{ind} (\tau(x_1), \ldots, \tau(x_n))$.
Zeige $\sigma(s) =_{E \cup L} \sigma(t)$ durch Noethersche Induktion nach $\succ_i$.

Voraussetzung hier $\sigma(y) \equiv \tau(y) \equiv y$ für alle $y \in$ Var$(s, t) - \{x_1, \ldots, x_n\}$.
Die x_i heißen **Induktionsvariablen**.

Verallgemeinerung der Problemstellung

Beachte

E enthält bedingte Gleichungen der Form
$u_1 = v_1, \ldots, u_n = v_n \Rightarrow s = t$.

Schreibweise

1. $u = v \Rightarrow s = t$ oder $s = t \Leftarrow u = v$, falls $u = (u_1, \ldots, u_n)$, $v = (v_1, \ldots, v_n)$.
2. $\sigma(u) = \sigma(v)$ statt $\sigma(u_1) = \sigma(v_1), \ldots, \sigma(u_n) = \sigma(v_n)$.

Beachte

$u = v \Rightarrow s = t$ gilt in $\mathcal{I} = \mathcal{I}(\mathsf{spec})$, falls für alle $\sigma \in C$-GSub$(u = v \Rightarrow s = t)$ gilt:

Gilt $\sigma(u) = \sigma(v)$ in $\mathcal{I}$, so gilt auch $\sigma(s) = \sigma(t)$ in $\mathcal{I}$.

Warnung: Satz 2.4 gilt nur für unbedingte Gleichungen.

Ab jetzt

$$ITh(\mathsf{spec}) = \{u = v \Rightarrow s = t \mid u = v \Rightarrow s = t \text{ gilt in } \mathcal{I}(\mathsf{spec})\}$$

Die Lemmata haben die Form $u = v \Rightarrow s = t$.

Die Behauptung hat die Form $u = v \Rightarrow s = t$.

3.1 Explizite Induktion

Sei sig, spec, C, L wie immer, $\mathcal{I} = \mathcal{I}(\mathsf{spec})$.

Ist $e \equiv u = v \Rightarrow s = t$ eine Gleichung so ist
$\sigma(e) \equiv \sigma(u) = \sigma(v) \Rightarrow \sigma(s) = \sigma(t)$.

Ansatz

Sei $e \equiv u = v \Rightarrow s = t$, zu zeigen ist $e \in ITh(\mathsf{spec})$. Wähle Noethersche Partialordnung $\succ_i$ auf C-GSub(e).

Zeige für alle $\sigma \in C$-GSub(e): Gilt $\tau(e)$ in $\mathcal{I}$ für alle $\tau \in C$-GSub(e) mit $\sigma \succ_i \tau$, so gilt $\sigma(e)$ in $\mathcal{I}$.

Dann gilt $e \in ITh(\mathsf{spec})$.

Fragen

a) Wie wählt man die Induktionsvariablen $x_1, \ldots, x_n \in \mathsf{Var}(e)$?
b) Wie wählt man die Induktionsordnung $\succ_i$?

Antwort

Das Definitionsschema der $f \in F - C$ hilft hier.

Bemerkungen

1. Bemerkung

Sei $\sigma \succ_i \tau$ gdw $(\sigma(x), \sigma(y)) >_{TT}^{lex} (\tau(x), \tau(y))$.

Dies entspricht der geschachtelten Induktion nach $>_{TT}$:
erst nach x, bei gleichem x nach y.

2. Bemerkung

Wir schreiben x statt $\sigma(x)$. Dies formalisiert „Sei x beliebig, aber fest".

3. Bemerkung

Häufig wird $\succ_i$ mit syntaktischen Ordnungen definiert, z. B. mit $>_{TT}$ oder einer LPO. Man kann aber auch semantische Ordnungen wählen, sie müssen nur Noethersch sein (z. B. die Länge von Listen).

4. Bemerkung

Wir benutzen auch die Fallunterscheidung als Beweismittel.

Ein Beispiel (1)

Sei $F = \{\text{true}, \text{false}, 0, s, \leq\}$, $C = \{\text{true}, \text{false}, 0, s\}$.

E: $0 \leq x = \text{true}$
$s(x) \leq 0 = \text{false}$
$s(x) \leq s(y) = x \leq y$

Dann ist C eine Konstruktormenge.

1. Behauptung $x \leq x = \text{true}$

Beweis: $x \equiv 0$: Es gilt $0 \leq 0 =_E \text{true}$.

$x \rightsquigarrow s(x)$: zu zeigen $s(x) \leq s(x) = \text{true}$.

Es gilt $s(x) \leq s(x) =_E x \leq x =_{IV} \text{true}$.

Beachte: Hier ist $\sigma \succ_i \tau$ gdw $\sigma(x) \equiv s(x) >_{TT} \tau(x) \equiv x$.

2. Behauptung $e \equiv x \leq 0 = \text{true} \Rightarrow x = 0$

Beweis: $x \equiv 0$: Es gilt $0 \leq 0 = \text{true} \Rightarrow 0 = 0$.

$x \rightsquigarrow s(x)$: Es gilt $s(x) \leq 0 \neq_E \text{true}$, also gilt
$s(x) \leq 0 = \text{true} \Rightarrow s(x) = 0$ in $\mathcal{I}$.

Ein Beispiel (2)

Schreibweise: t statt $t =$ true, falls t von der Sorte BOOL ist.

3. Behauptung

$e \equiv x \leq y =$ true$, y \leq z =$ true $\Rightarrow x \leq z =$ true

Wähle die Induktionsordnung $\succ_i$ als
$\sigma \succ_i \tau \text{ gdw } (\sigma(x), \sigma(y), \sigma(z)) >_{TT}^{lex} (\tau(x), \tau(y), \tau(z))$.

1. $\sigma(x) \equiv 0$: Wegen $0 \leq z =$ true gilt $\sigma(e)$ in $\mathcal{I}$.

2. $\sigma(x) \equiv s(x)$: Zu zeigen $s(x) \leq y, y \leq z \Rightarrow s(x) \leq z$.

2.1 $\sigma(y) \equiv 0$: Wegen $s(x) \leq 0 \neq_E$ true gilt $\sigma(e)$ in $\mathcal{I}$.

2.2 $\sigma(y) \equiv s(y)$:
Zu zeigen $s(x) \leq s(y)$, $s(y) \leq z \Rightarrow s(x) \leq z$.

2.2.1 $\sigma(z) \equiv 0$: Wegen $s(y) \leq 0 \neq_E$ true gilt $\sigma(e)$ in $\mathcal{I}$.

2.2.2 $\sigma(z) \equiv s(z)$:
Zu zeigen $s(x) \leq s(y), s(y) \leq s(z) \Rightarrow s(x) \leq s(z)$.
Dies ist E-gleich zu $\tau(e) \equiv x \leq y, y \leq z \Rightarrow x \leq z$.
Dies gilt nach IV mit $\tau =$ id, also gilt $\sigma(e)$ in $\mathcal{I}$.

Ein Beispiel (3)

4. Behauptung $x \leq y =$ false $\Rightarrow y \leq x =$ true.

Beweis: Übung.

5. Behauptung $x \leq y =$ true $\vee\ x \leq y =$ false.

Beweis: Übung.

Beachte

a) $t_1 \neq_E t_2$ steht hier für $\mathcal{I} \models t_1 \neq t_2$ mit $\mathcal{I} = \mathcal{I}$(spec).

b) Es gilt $\mathcal{I} \models u = v \Rightarrow s = t$ genau dann, wenn für jede Grundsubstitution σ gilt: $\sigma(u) \neq_E \sigma(v)$ oder $\sigma(s) =_E \sigma(t)$.

c) Also: Aus $u \neq_E v$ folgt $\mathcal{I} \models u = v \Rightarrow s = t$.

Beachte zum Nachweis von $t_1 =_E t_2$ bzw. $t_1 \neq_E t_2$:

Richten von E ergebe konvergentes R.

Also gilt $t_1 =_E t_2 \text{ gdw } t_1 \downarrow_R t_2$.
$t_1 \neq_E t_2$ falls $t_i \xrightarrow{*}_R c_i$, $c_1 \not\equiv c_2$ irreduzible Grundterme.

Ein Beispiel (4)

Sei

E erweitert um („Sortieren durch Einfügen")

ins$(x, \text{nil}) = x.\text{nil}$
ins$(x, y.l) = x.y.l \Leftarrow x \leq y = \text{true}$
ins$(x, y.l) = y.\text{ins}(x, l) \Leftarrow x \leq y = \text{false}$
sorted(nil)=true
sorted$(x.\text{nil})$=true
sorted$(x.y.l) = \text{sorted}(y.l) \Leftarrow x \leq y = \text{true}$
sorted$(x.y.l) = \text{false} \Leftarrow x \leq y = \text{false}$

Beachte E gerichtet zu R ist konfluent und terminierend.

$C = \{\text{true}, \text{false}, 0, s, \text{nil}, .\}$ ist eine Konstruktormenge.

Behauptung $e \equiv \text{sorted}(l) \Rightarrow \text{sorted}(\text{ins}(x, l))$

Das Definitionsschema für sorted legt nahe:

a) Induktion nach l mit $>_{TT}$ als Induktionsordnung.
b) 3 Fälle: $l \equiv \text{nil}$ und $l \equiv y.\text{nil}$ als Basisfälle und $l \equiv y.z.l'$ für den Induktionsschritt.

Ein Beispiel (5)

1. $\sigma(l) \equiv \text{nil}$:
Wegen sorted(ins$(x, \text{nil})) =_E$ sorted$(x.\text{nil}) =_E$ true gilt $\sigma(e)$.

2. $\sigma(l) \equiv y.\text{nil}$:
Wir zeigen sorted(ins$(x, y.\text{nil})) =_E$ true, dann gilt $\sigma(e)$.
Beachte, dass $x \leq y = \text{true} \vee x \leq y = \text{false}$ ein Lemma ist.
Wir machen eine Fallunterscheidung.

2.1 $x \leq y = \text{true}$: Es gilt
sorted(ins$(x, y.\text{nil})) =_E$ sorted$(x.y.\text{nil}) =_E$ sorted$(y.\text{nil}) =_E$ true.

2.2 $x \leq y = \text{false}$, also $y \leq x = \text{true}$ (siehe oben):
sorted(ins$(x, y.\text{nil})) =_E$ sorted$(y.x.\text{nil}) =_E$ true

3. $\sigma(l) \equiv y.z.l$:
Zu zeigen ist $\sigma(e) \equiv \text{sorted}(y.z.l) \Rightarrow \text{sorted}(\text{ins}(x, y.z.l))$.
Wir benutzen wieder die Fallunterscheidung.

Ein Beispiel (6)

3.1 $y \leq z =$ false: Wegen sorted$(y.z.l) \neq_E$ true gilt $\sigma(e)$.

3.2 $y \leq z =$ true: Es gilt sorted$(y.z.l) =_E$ sorted$(z.l)$.

3.2.1 $x \leq y =$ true:
$\sigma(e)$ reduziert sich mit E zu sorted$(z.l) \Rightarrow$ sorted$(z.l)$.
Dies gilt in I(spec).

3.2.2 $x \leq y =$ false, also $y \leq x =$ true.

3.2.2.1 $x \leq z =$ true:
Umschreiben von $\sigma(e)$ mit E liefert sorted$(z.l) \Rightarrow$ sorted$(z.l)$.
Dies gilt in I(spec).

3.2.2.2 $x \leq z =$ false:
Umschreiben von $\sigma(e)$ liefert sorted$(z.l) \Rightarrow$ sorted(ins$(x, z.l)$).
(denn sorted(ins$(x, y.z.l)) =_E$ sorted$(y.$ins$(x, z.l)) =_E$ sorted$(y.z.$ins$(x.l)) =_E$ sorted$(z.$ins$(x, l)) =_E$ sorted(ins$(x, z.l)$)).
Dies gilt nach IV.

Flexible Wahl der Induktionsordnung

Beachte

Einzige Voraussetzung an die Induktionsordnung ist: $\succ_i$ ist Noethersch.

Sei $F = \{0, s, \max\}$, $C = \{0, s\}$

E:
$$\max(x, 0) = x$$
$$\max(0, y) = y$$
$$\max(s(x), s(y)) = s(\max(x, y))$$

Dann ist C eine Konstruktormenge.

Behauptung

Es gilt $\max(x, y) = \max(y, x)$ in $\mathcal{I}$.

Beweis:

Sei $\sigma \succ_i \tau$ gdw $\{\sigma(x), \sigma(y)\} \gg_{TT} \{\tau(x), \tau(y)\}$.

1. Fall: $\sigma = \{y \leftarrow 0\}$.

Es gilt: $\max(x, 0) =_E x =_E \max(0, x)$.

2. Fall: $\sigma = \{x \leftarrow 0\}$.

Es gilt: $\max(0, y) =_E y =_E \max(y, 0)$.

3. Fall: $\sigma = \{x \leftarrow s(x), y \leftarrow s(y)\}$.

Es gilt: $\max(s(x), s(y)) =_E s(\max(x, y)) =_{IV} s(\max(y, x)) =_E \max(s(y), s(x))$.

Semantische Induktionsordnung

Beispiel

Sortieren durch Mischen: msort.

R: split1(nil) $\longrightarrow$ nil
split1(x.nil) $\longrightarrow$ x.nil
split1($x.y.l$) $\longrightarrow$ x.split1(l)
split2(nil) $\longrightarrow$ nil
split2(x.nil) $\longrightarrow$ nil
split2($x.y.l$) $\longrightarrow$ y.split2(l)
merge(nil, nil) $\longrightarrow$ nil
merge($x.l_1$, nil) $\longrightarrow$ $x.l_1$
merge(nil, $y.l_2$) $\longrightarrow$ $y.l_2$
merge($x.l_1, y.l_2$) $\longrightarrow$ x.merge($l_1, y.l_2$) $\Leftarrow x \leq y =$ true
merge($x.l_1, y.l_2$) $\longrightarrow$ y.merge($x.l_1, l_2$) $\Leftarrow x \leq y =$ false
msort(nil) $\longrightarrow$ nil
msort(x.nil) $\longrightarrow$ x.nil
msort($x.y.l$) $\longrightarrow$ merge(msort(split1($x.y.l$)), msort(split2($x.y.l$)))

Behauptung: sorted(msort(l)) = true.

Beweis-Idee:
Setze le(l)=Länge von l und zeige

le($x.y.l$) $>$ le(split1($x.y.l$))
le($x.y.l$) $>$ le(split2($x.y.l$))
Setze $\sigma \succ_i \tau$ gdw le($\sigma(l)$) $>$ le($\tau(l)$)

Also: Induktion nach der Länge von l.

3.2 Die Coverset-Methode

Sei sig, spec = (sig, E), $C \subseteq F$ wie immer.
$e \equiv u = v \Rightarrow s = t$.

Frage

Wie findet man eine Induktionsordnung $\succ_i$ und ein Induktionsschema für den Beweis zu $e \in ITh$(spec)?

Idee dazu

Wähle Induktionsschema nach dem Definitionsschema eines „geeigneten“ $f \in F - C$. Dies Induktionsschema wird durch ein „Coverset“ beschrieben.

Benutze $>_{TT}$ zur Definition von $\succ_i$.

Falls spec = (sig, R) und R simplifizierend bezüglich $\succ$, so benutze evtl. auch $\succ$ zur Definition von $\succ_i$. Beispiel: Sei $t \in$ Term(F, V) gegeben. Setze $\sigma \succ_i \tau$ gdw $\sigma(t) \succ \tau(t)$.

Ein motivierendes Beispiel (1)

Sei $F = \{0, s, \text{double}, \text{half}, \text{true}, \text{false}, \text{even}\}$,
$C = \{0, s, \text{true}, \text{false}\}$.

R:

$\text{even}(0) \longrightarrow \text{true}$	$\text{half}(0) \longrightarrow 0$
$\text{even}(s(0)) \longrightarrow \text{false}$	$\text{half}(s(0)) \longrightarrow 0$
$\text{even}(s(s(x))) \longrightarrow \text{even}(x)$	$\text{half}(s(s(x))) \longrightarrow s(\text{half}(x))$
$\text{double}(0) \longrightarrow 0$	
$\text{double}(s(x)) \longrightarrow s(s(\text{double}(x)))$	

$e \equiv \text{even}(x) = \text{true} \Rightarrow \text{double}(\text{half}(x)) = x \in ITh(\text{spec})$.

Dann: C ist eine Konstruktormenge.

Idee

Induktion nach dem Definitionsschema von even.

1. Fall: $\sigma_1(x) \equiv 0$ Basisfall.
2. Fall: $\sigma_2(x) \equiv s(0)$ Basisfall.
3. Fall: $\sigma_3(x) \equiv s(s(x))$ reduziert $\sigma_3(e)$ auf $\tau(e)$ mit $\tau(x) \equiv x$.

Also

$M = \{0, s(0), s(s(x))\}$ ist eine „Überdeckung" von $\text{Term}(C)$.

$C_{\text{even}} = \{\langle 0, \varnothing\rangle, \langle s(0), \varnothing\rangle, \langle s^2(x), \{x\}\rangle\}$ ist ein „Coverset" für even $\in F$.

Ein motivierendes Beispiel (2)

C_{even} liefert das Induktionsschema

(1) $\Rightarrow \sigma_1(e)$
(2) $\Rightarrow \sigma_2(e)$
(3) $\tau(e) \Rightarrow \sigma_3(e)$

und die Induktionsordnung

$\sigma' \succ_i \sigma \text{ gdw } \sigma'(x) >_{TT} \sigma(x)$

Es gilt

Zu (1):
$\text{even}(0) = \text{true} \Rightarrow \text{double}(\text{half}(0)) = 0 \in ITh(\text{spec})$.

Zu (2):
$\text{even}(s(0)) = \text{true} \Rightarrow \text{double}(\text{half}(s(0))) = s(0) \in ITh(\text{spec})$
denn $\text{even}(s(0)) \neq_R \text{true}$.

Zu (3): Es reduziert sich

$\text{even}(s^2(x)) = \text{true} \Rightarrow \text{double}(\text{half}(s^2(x))) = s^2(x)$.

zu

$\text{even}(x) = \text{true} \Rightarrow s^2(\text{double}(\text{half}(x))) = s^2(x)$.

Dies gilt, denn nach IV gilt

$\tau(e) \equiv \text{even}(x) = \text{true} \Rightarrow \text{double}(\text{half}(x)) = x$.

Frage

Wie kommt man (automatisch) zu dem Induktionsschema?

Konstruktor-basierte Spezifikationen (1)

Beachte

Stärkere Voraussetzungen an die Spezifikationen erlauben effizientere Algorithmen.

Idee

spec = (sig, R), sig = (S, F, α), $F = C \cup D$ mit $C \cap D = \emptyset$.

Das grund-konvergente R definiert die $f \in D$ (definierte Operatoren) eventuell partiell über den Konstruktortermen in Term(C). Die $c \in C$ heißen **syntaktische Konstruktoren**.

Ab jetzt

Sei $F = C \cup D$ mit $C \cap D = \emptyset$.

Sei $f \in D$. Eine f-**Definition** ist eine endliche Menge bedingter Regeln der Form $l \to r \Leftarrow u = v$ mit
(1) $l \equiv f(t_1, \ldots, t_n)$ mit $t_i \in$ Term(C, V),
(2) $r, u_i, v_i \in$ Term(F, V),
(3) Var$(r, u_i, v_i) \subseteq$ Var(l) und
(4) $l \succ r, u_i, v_i$ für eine Simplifikationsordnung $\succ$.

Konstruktor-basierte Spezifikationen (2)

Definition 3.1

Sei sig = (S, F, α), $F = C \cup D$ und $C \cap D = \emptyset$.

Eine **konstruktor-basierte Spezifikation** ist spec = (sig, R) mit $R = R_C \cup \bigcup_{f \in D} R_f$.

Dabei sind die R_f f-Definitionen und R_C ist ein Regelsystem über sig$_c = (S, C, \alpha_c)$. Weiter ist R simplifizierend bezüglich einer Reduktionsordnung $\succ$.

Beachte

Ist $R_C = \emptyset$, so sind die Konstruktoren in C **frei**.

In der Praxis sind viele Spezifikationen konstruktor-basiert.

Beispiele

a) Die Spezifikationen zu ins und double/half sind konstruktor-basiert.
b) Die Spezifikation der Ackermann-Funktion ist konstruktor-basiert.

$$\begin{aligned} R\colon\quad & a(0, y) \to s(y) \\ & a(s(x), 0) \to a(x, s(0)) \\ & a(s(x), s(y)) \to a(x, a(s(x), y)) \end{aligned}$$

Coversets

Definition 3.2

Sei spec $= (\text{sig}, R)$ eine konstruktor-basierte Spezifikation.

sig $= (S, F, \alpha)$ und $C \subseteq F$.

Sei $l \rightarrow r \Leftarrow u = v$ in R_f und $l \equiv f(t_1, \ldots, t_n)$. Sei M die Menge der Argument-n-Tupel von f in r. Dann heißt

$$\langle (t_1, \ldots, t_n), M, u = v \rangle$$

das **Überdeckungstripel** zu $l \rightarrow r \Leftarrow u = v$.

Sei C_f die Menge der Überdeckungstripel zu den Regeln in R_f. Dann heißt C_f ein **Coverset** für f.

Beispiel Ackermann-Funktion

C_a besteht aus

$\langle (0, y), \varnothing, \varnothing \rangle \qquad \langle (s(x), 0), \{(x, s(0))\}, \varnothing \rangle$

$\langle (s(x), s(y)), \{(x, a(s(x), y)), (s(x), y)\}, \varnothing \rangle$

Induktionsschemata (1)

Sei

spec $= (\text{sig}, R)$ konstruktor-basiert.

$e \equiv u = v \Rightarrow s = t$.

Zu zeigen: e gilt in $\mathcal{I} = \mathcal{I}(\text{spec})$.

Beachte

Ein Induktionsschema wird mit Hilfe eines Teilterms t_0 definiert. Dazu gibt es verschiedene Möglichkeiten. Mit einer guten Heuristik ist t_0 zu wählen.

Hier

Definition eines Induktionsschemas zu $t_0 \equiv f(x_1, \ldots, x_n)$ mit $f \in F - C$.

Beispiel

$e \equiv \text{even}(x) = \text{true} \Rightarrow \text{double}(\text{half}(x)) = x$.

Wähle $t_0 \equiv \text{even}(x)$, also $f = \text{even}$.

Induktionsschemata (2)

Definition 3.3

Sei $c = \langle (s_1, \ldots, s_n), M, u' = v' \rangle$ ein Überdeckungstripel zu f.

Sei $M = \{(t_1^i, \ldots, t_n^i) \mid 1 \leq i \leq k\}$, $e \equiv u = v \Rightarrow s = t$ und $e/\pi \equiv f(x_1, \ldots, x_n)$. Dies definiert

$$\sigma = \{x_j \leftarrow s_j \mid 1 \leq j \leq n\} \qquad \tau_i = \{x_j \leftarrow t_j^i \mid 1 \leq j \leq n\}$$

Dann ist

$$[u' = v' \wedge \bigwedge_{i=1}^{k} \tau_i(e)] \Rightarrow \sigma(e)$$

das **Induktions-Teilziel** zum Überdeckungstripel c. Die Menge dieser Teilziele bildet das **Induktionsschema** zu spec, e, $f(x_1, \ldots, x_n)$.

Beachte

a) Wir setzen voraus, dass e und $\langle (s_1, \ldots, s_n), M, u' = v' \rangle$ Variablen-disjunkt sind.

b) Jedes Überdeckungstripel in C_f erzeugt genau ein Teilziel.

Beispiel: Ackermann-Funktion

Sei

$F = \{0, s, a, +, \leq, \text{true}, \text{false}\}$, $C = \{0, s, \text{true}, \text{false}\}$.

R: $a(0, y) \to s(y)$
$a(s(x), 0) \to a(x, s(0))$
$a(s(x), s(y)) \to a(x, a(s(x), y))$
und Definitionen von $+$, $\leq$ wie früher

$e \equiv x' + y' \leq a(x', y') = \text{true}$.

$\succ$ die LPO zur Präzedenz $a >_F s >_F 0$

Also

$$C_a = \{\langle (0, y), \emptyset, \emptyset \rangle, \langle (s(x), 0), \{(x, s(0))\}, \emptyset \rangle, \langle (s(x), s(y)), \{(x, a(s(x), y)), (s(x), y)\}, \emptyset \rangle\}$$

Wähle die Induktionsordnung
$\sigma \succ_i \tau$ gdw $\sigma(a(x, y)) \succ \tau(a(x, y))$

Dies liefert die Teilziele (ohne „= true“)

1. $\Rightarrow 0 + y \leq a(0, y)$.
2. $x + s(0) \leq a(x, s(0)) \Rightarrow s(x) + 0 \leq a(s(x), 0)$.
3. $s(x)+y \leq a(s(x), y), x+a(s(x), y) \leq a(x, a(s(x), y))$
$\Rightarrow s(x) + s(y) \leq a(s(x), s(y))$

Beispiel: Sortieren durch Einfügen

Sei $F = \{0, s, \mathsf{nil}, ., \mathsf{ins}, \mathsf{sorted}, \leq, \mathsf{true}, \mathsf{false}\}$,
$C = \{0, s, \mathsf{true}, \mathsf{false}, \mathsf{nil}, .\}$

$$
\begin{array}{lll}
R: & \mathsf{sorted}(\mathsf{nil}) \longrightarrow \mathsf{true} & \\
 & \mathsf{sorted}(x.\mathsf{nil}) \longrightarrow \mathsf{true} & \\
 & \mathsf{sorted}(x.y.l) \longrightarrow \mathsf{sorted}(y.l) & \Leftarrow x \leq y = \mathsf{true} \\
 & \mathsf{sorted}(x.y.l) \longrightarrow \mathsf{false} & \Leftarrow x \leq y = \mathsf{false}
\end{array}
$$

und Definitionen von ins, $\leq$ wie früher

$e \equiv \mathsf{sorted}(l') = \mathsf{true} \Rightarrow \mathsf{sorted}(\mathsf{ins}(x', l')) = \mathsf{true}$

Das Coverset $C_f = C_{\mathsf{sorted}}$ ist

1. $\langle \mathsf{nil}, \emptyset, \emptyset \rangle$
2. $\langle x.\mathsf{nil}, \emptyset, \emptyset \rangle$
3. $\langle x.y.l, \{y.l\}, x \leq y = \mathsf{true} \rangle$
4. $\langle x.y.l, \emptyset, x \leq y = \mathsf{false} \rangle$

Dies liefert das Induktionsschema (ohne „$=$ true")

1. $\Rightarrow [\mathsf{sorted}(\mathsf{nil}) \Rightarrow \mathsf{sorted}(\mathsf{ins}(x', \mathsf{nil}))]$
2. $\Rightarrow [\mathsf{sorted}(x.\mathsf{nil}) \Rightarrow \mathsf{sorted}(\mathsf{ins}(x', x.\mathsf{nil}))]$
3. $x \leq y = \mathsf{true} \wedge (\mathsf{sorted}(y.l) \Rightarrow \mathsf{sorted}(\mathsf{ins}(x', y.l)))$
   $\Rightarrow [\mathsf{sorted}(x.y.l) \Rightarrow \mathsf{sorted}(\mathsf{ins}(x', x.y.l))]$
4. $x \leq y = \mathsf{false} \Rightarrow [\mathsf{sorted}(x.y.l) \Rightarrow \mathsf{sorted}(\mathsf{ins}(x', x.y.l))]$

Bemerkung

Sei

Gegeben: $\mathsf{spec} = (\mathsf{sig}, R)$, $e \equiv u = v \Rightarrow s = t$.

Zu zeigen: e gilt in $\mathcal{I} = \mathcal{I}(\mathsf{spec})$.

Ziel

Sind die Teilziele gültig in $\mathcal{I}$, so ist auch e gültig in $\mathcal{I}$.

Beachte

Dazu benötigt man

a) die Vollständigkeit: Jede Grundinstanz $\sigma(e)$ wird durch einen Fall im Induktionsschema abgedeckt.

b) die Korrektheit: Es gibt eine Noethersche Ordnung (Induktionsordnung) nach der in jedem Teilziel die Voraussetzung kleiner ist als die Behauptung.

Zur Korrektheit

Sei spec $=$ (sig, R)
R bezüglich der Simplifikationsordnung $\succ$ simplifizierend.
$e/\pi = f(x_1, \ldots, x_n)$.
$l \to r \Leftarrow u = v$ in R, $f = \mathsf{top}(l)$.

Sei $\langle (s_1, \ldots, s_n), M, u = v\rangle \in C_f$, $(t_1^i, \ldots, t_n^i) \in M$.

Dies erzeugt das Teilziel

$$[u = v \wedge \bigwedge_{i=1}^{k} \tau_i(e)] \Rightarrow \sigma(e)$$

mit $\sigma(x_j) \equiv s_j$ und $\tau_i(x_j) \equiv t_j^i$ und es gilt
$l \equiv \sigma(f(x_1, \ldots, x_n))$ und die $\tau_i(f(x_1, \ldots, x_n))$ sind Teilterme von r.

Also gilt $\sigma(f(x_1, \ldots, x_n)) = l \succ r \succeq \tau_i(f(x_1, \ldots, x_n))$.

Definition 3.4

$\sigma \succ_i \tau \ \mathrm{gdw}\ \sigma(f(x_1, \ldots, x_n)) \succ \tau(f(x_1, \ldots, x_n))$ ist die durch spec, e, $f(x_1, \ldots, x_n)$ definierte Induktionsordnung.

Beachte

$\succ_i$ ist eine Noethersche Ordnung auf C-GSub(e).

Die Bedingung für die Korrektheit ist automatisch erfüllt.

Zur Vollständigkeit (1)

Sei spec $=$ (sig, R) und $C \subseteq F$ wie stets.

Definition 3.5

Das Coverset C_f für $f(x_1, \ldots, x_n)$, $f \in F - C$ heißt **vollständig**, falls es zu jedem $\hat{t} = (t_1, \ldots, t_n) \in \mathsf{Term}(C)^n$ ein $\langle \hat{s}, M, u = v\rangle \in C_f$ und ein σ gibt mit
$\sigma(\hat{s}) \equiv \hat{t}$ und $\sigma(u) =_R \sigma(v)$.

Beachte

a) Ist C_f für alle $f \in F$ vollständig, so ist jeder Grundterm der Form $f(t_1, \ldots, t_n)$ in R reduzierbar, es gilt also $f(t_1, \ldots, t_n) \xrightarrow{*}_R t_0$ für ein $t_0 \in \mathsf{Term}(C)$. Also ist f durch R total definiert.

b) Ist f durch R total definiert, so ist C_f vollständig.

c) In den bisherigen Beispielen war stets C_f vollständig.

d) Den Nachweis, dass f total definiert ist, kann man auch durch Induktion zeigen: Definiere das Prädikat $\mathrm{def}(t)$ auf $\mathcal{I} = \mathcal{I}(\mathsf{spec})$ durch $\mathcal{I} \models \mathrm{def}(t)\ \mathrm{gdw}\ \exists t' \in \mathsf{Term}(C) : t =_R t'$.
Zeige dann $\mathcal{I} \models \mathrm{def}(f(x_1, \ldots, x_n))$.

Zur Vollständigkeit (2)

Sei spec $=$ (sig, R)
$e \equiv u = v \Rightarrow s = t$ und $e/\pi \equiv f(x_1, \ldots, x_n)$.

Lemma 3.1

Sei das Coverset C_f vollständig. Zu jedem $\varrho \in C$-GSub(e) gibt es ein Teilziel

$$[u = v \wedge \bigwedge_{i=1}^{k} e_i] \Rightarrow e_0$$

im Induktionsschema zu spec, e, f und ein σ mit
$\sigma(u) =_R \sigma(v)$ und $\varrho(e) \equiv \sigma(e_0)$.

Beweis: Sei $\varrho(\hat{x}) \equiv \hat{t} = (t_1, \ldots, t_n)$. Da C_f vollständig ist, gibt es σ und $\langle \hat{s}, M, u = v \rangle \in C_f$ mit $\sigma(\hat{s}) \equiv \hat{t}$ und $\sigma(u) =_R \sigma(v)$. Dies $\langle \hat{s}, M, u = v \rangle$ erzeuge

$$[u = v \wedge \bigwedge_{i=1}^{k} e_i] \Rightarrow e_0.$$

Sei $\sigma_0(\hat{x}) = \hat{s}$, also $\varrho(\hat{x}) \equiv \hat{t} \equiv \sigma(\hat{s}) \equiv \sigma(\sigma_0(\hat{x}))$. Dann ist $\varrho(e) \equiv \sigma(\sigma_0(e)) \equiv \sigma(e_0)$.

Beachte

In $\langle \hat{s}, M, u = v \rangle$ kann sowohl M als auch $u = v$ leer sein. Ist $u = v$ leer, so entfällt im Teilziel $u = v \wedge \hat{e} \Rightarrow e_0$ die Vorbedingung „$u = v$“. Ist auch M leer, so entfällt „$u = v \wedge \hat{e}$“ .

Das Ergebnis

Sei spec $=$ (sig, R) konstruktor-basiert
sig $= (S, F, \alpha)$, $C \subseteq F$
R simplifizierend bezüglich $\succ$
$e \equiv u' = v' \Rightarrow s = t$ und $e/\pi \equiv f(x_1, \ldots, x_n)$

Satz 3.1.

Sei C_f vollständig und jedes Teilziel

$$u = v \wedge \hat{e} \Rightarrow e_0$$

aus dem Induktionsschema zu spec, e, $f(x_1, \ldots, x_n)$ gelte in $\mathcal{I}$(spec). Dann gilt auch e in $\mathcal{I}$(spec).

Beweis zu Satz 3.1

Sei $\varrho \in C$-GSub(e); zu zeigen ist, dass $\varrho(e)$ in $\mathcal{I}(\text{spec})$ gilt.

Wir führen Noethersche Induktion nach $\succ_i$.

Zu ϱ gibt es nach Lemma 3.1 ein Teilziel

$$[u = v \wedge \bigwedge_{j=1}^{k} \tau_j(e)] \Rightarrow \sigma(e)$$

und ein σ' mit $\varrho = \sigma' \circ \sigma$ auf Var(e) und $\sigma'(u) =_R \sigma'(v)$.

Es ist $\sigma \succ_i \tau_j$, also auch $\varrho \succ_i \varrho_j$ mit $\varrho_j = \sigma' \circ \tau_j$.

Also gilt nach IV $\varrho_j(e)$ in $\mathcal{I}(\text{spec})$ für $j = 1, \ldots, k$.

Da das Teilziel in $\mathcal{I}(\text{spec})$ gilt und wegen $\sigma'(u) =_R \sigma'(v)$ gilt $\varrho(e) \equiv \sigma'(\sigma(e))$ in $\mathcal{I}(\text{spec})$.

Ein Beweis zur Ackermann-Funktion (1)

Sei $F = \{\text{true}, \text{false}, 0, s, +, a, \leq\}$, $C = \{0, s, \text{true}, \text{false}\}$.

$$
\begin{array}{lll}
R: & x + 0 \to x & a(0,y) \to s(y) \\
 & x + s(y) \to s(x+y) & a(s(x),0) \to a(x, s(0)) \\
 & 0 \leq y \to \text{true} & a(s(x),s(y)) \to a(x, a(s(x),y)) \\
 & s(x) \leq 0 \to \text{false} & \\
 & s(x) \leq s(y) \to x \leq y &
\end{array}
$$

$e \equiv s(x+y) \leq a(x,y)$

Behauptung

e gilt in $\mathcal{I}(\text{spec})$.

Es gelten folgende Lemmata in $\mathcal{I}$ (o. Beweis)

L1: $0 + y = y$
L2: $s(x) + y = s(x+y)$
L3: $x + (y+z) = (x+y)+z$
L4: $x \leq y = \text{true}, y \leq z = \text{true} \Rightarrow x \leq z = \text{true}$
L5: $x \leq y = \text{true} \Rightarrow s(x) \leq s(y) = \text{true}$
L6: $x \leq y = \text{true} \Rightarrow z + x \leq z + y = \text{true}$
L7: $x \leq x = \text{true}$

Es gilt

$$
\begin{aligned}
C_a = \{ & \langle (0,y), \varnothing, \varnothing \rangle, \langle (s(x),0), \{x, s(0)\}, \varnothing \rangle, \\
& \langle (s(x),s(y)), \{(x, a(s(x),y)), (s(x),y)\}, \varnothing \rangle \}
\end{aligned}
$$

Ein Beweis zur Ackermann-Funktion (2)

Das Coverset C_a liefert die **Teilziele** (ohne „$=$ true")

1. $\Rightarrow s(0+y) \leq a(0,y)$
2. $s(x+s(0)) \leq a(x,s(0)) \Rightarrow s(s(x)+0) \leq a(s(x),0)$
3. $s(s(x)+y) \leq a(s(x),y)$,
   $s(x+a(s(x),y)) \leq a(x,a(s(x),y))$
   $\Rightarrow s(s(x)+s(y)) \leq a(s(x),s(y))$

Zu 1. Umschreiben mit R und L1 liefert

$\Rightarrow s(y) \leq s(y)$
Dies gilt in $\mathcal{I}$.

Zu 2. Umschreiben mit R und L2 liefert

$s^2(x) \leq a(x,s(0)) \Rightarrow s^2(x) \leq a(x,s(0))$
Dies gilt in $\mathcal{I}$.

Zu 3. Umschreiben mit R und L2 liefert

$s^2(x+y) \leq a(s(x),y)$,
$s(x+a(s(x),y)) \leq a(x,a(s(x),y))$
$\Rightarrow s^3(x+y) \leq a(x,a(s(x),y))$

Die Behauptung folgt mit den Lemmata L1-L7 aus der Voraussetzung.

Ein Beweis zur Ackermann-Funktion (3)

Sei

$s^2(x+y) \leq a(s(x),y)$
$s(x+a(s(x),y)) \leq a(x,a(s(x),y))$

Dann

$x+s^2(x+y) \leq x+a(s(x),y)$	nach L6
$s(x+s^2(x+y)) \leq s(x+a(s(x),y))$	nach L5
$s(x+s^2(x+y)) \leq a(x,a(s(x),y))$	nach L4
$s^3(x+(x+y)) \leq a(x,a(s(x),y))$	nach R
$x+y \leq x+(x+y)$	nach L6 und R
$s^3(x+y) \leq s^3(x+(x+y))$	nach L5
$s^3(x+y) \leq a(x,a(s(x),y))$	nach L4

Beachte

Im Beweis zu e

- sind die Variablen in e „beliebig aber fest",
- können Lemmata mit beliebiger Instanziierung der Variablen benutzt werden.

Zur Wahl der Lemmata

Beachte

Die Wahl der „richtigen" Lemmata ist zum Beweis der Gültigkeit von e in $\mathcal{I}(\text{spec})$ von zentraler Bedeutung.

Frage

Wie findet man sie?

Antwort

1. Das ist i. Allg. sehr schwer!
2. Häufig helfen Heuristiken weiter:
   - Wähle das, was zum Beweis gerade benötigt wird, als Hypothese.
   - Beweise die Hypothese, dann ist sie ein Lemma.

3.3 Induktionsbeweise durch Konsistenzprüfung

Bezeichnung

Dieses Verfahren läuft in der Literatur auch unter den Namen „Induktionslose Induktion" und „Proof by Consistency".

Vorteil des Verfahrens

Man benötigt kein Induktionsschema.

Nachteil des Verfahrens

Es hält sehr häufig nicht.

Beschränkung hier

Sowohl die Spezifikation als auch die Beweisziele sind unbedingt.

Konsistenz

Sei spec $= (\text{sig}, E)$
sig $= (S, F, \alpha)$
$E \subseteq \text{Term}(F, V) \times \text{Term}(F, V)$
$G \subseteq \text{Term}(F, V) \times \text{Term}(F, V)$

Ziel

Zeige $G \subseteq ITh(\text{spec})$.

Definition 3.6

G heißt **konsistent** (in spec), falls gilt

$=_E \; = \; =_{E \cup G}$ auf $\text{Term}(F)$.

Beachte

Es gilt stets $=_E \subseteq =_{E \cup G}$.

Also: G ist konsistent, falls für alle $s, t \in \text{Term}(F)$ gilt:

Aus $s =_{E \cup G} t$ folgt $s =_E t$.

Bedeutung der Konsistenz

Lemma 3.2

Es gilt

$G \subseteq ITh(\text{spec})$ gdw G ist konsistent in spec.

Beweis:

„$\curvearrowleft$": Sei $u = v$ in G, $\sigma \in \text{GSub}(u = v)$.

Es gilt $\sigma(u) =_G \sigma(v)$, also $\sigma(u) =_{E \cup G} \sigma(v)$, also $\sigma(u) =_E \sigma(v)$, da G konsistent ist.

Also $u = v \in ITh(\text{spec})$.

Also $G \subseteq ITh(\text{spec})$.

„$\curvearrowright$": Sei $s, t \in \text{Term}(F)$ und $s =_{E \cup G} t$.

Jeder G-Schritt $t_1 \vdash\!\dashv_G t_2$ im Beweis zu $s =_{E \cup G} t$ kann durch einen E-Beweis $t_1 =_E t_2$ ersetzt werden, wegen $G \subseteq ITh(\text{spec})$.

Also gilt $s =_E t$. Also ist G konsistent.

Test auf Konsistenz (1)

Sei spec $= (\text{sig}, E)$
sig $= (S, F, \alpha)$, G wie immer
R Regelsystem über sig

Bezeichnung

R heißt **Regelsystem zu** E, falls $=_R \, = \, =_E$ gilt.

$\text{GIrr}(R) = \{t \in \text{Term}(F) \mid t \text{ ist } R\text{-irreduzibel}\}$

Lemma 3.3

Seien R und R' grundkonvergente Regelsysteme zu E bzw. $E \cup G$.
Gilt $\text{GIrr}(R) = \text{GIrr}(R')$, so ist G konsistent.

Beweis: Sei $s, t \in \text{Term}(F)$, $t_R \equiv t\downarrow_R$, $t_{R'} \equiv t\downarrow_{R'}$.

a) Es gilt stets $t_R \equiv t_{R'}$:
Es gilt $t_R =_E t =_{E\cup G} t_{R'}$, also $t_R =_{R'} t_{R'}$, also $t_R \downarrow_{R'} t_{R'}$.
Also gilt $t_R \equiv t_{R'}$ wegen $\text{GIrr}(R) = \text{GIrr}(R')$.

b) Aus $s =_{E\cup G} t$ folgt $s =_E t$:
Aus $s =_{E\cup G} t$ folgt $s_{R'} \equiv t_{R'}$. Nach a) gilt $s_R \equiv t_R$, also $s =_R t$, also $s =_E t$.

c) G ist konsistent nach b).

Grundreduzierbarkeit

Sei

R Regelsystem über sig $= (S, F, \alpha)$.

Definition 3.7

$t \in \text{Term}(F, V)$ heißt **grundreduzierbar** in R, falls für jede Grundinstanz $\sigma(t)$ (d. h. jedes $\sigma \in \text{GSub}(t)$) der Term $\sigma(t)$ R-reduzierbar ist.

Beispiel

$$R: \quad \begin{array}{ll} x + 0 \to x & \text{app}(\text{nil}, l_2) \to l_2 \\ x + s(y) \to s(x+y) & \text{app}(x.l_1, l_2) \to x.\text{app}(l_1, l_2) \end{array}$$

Es sind $t_1 \equiv x + y$ und $t_2 \equiv \text{app}(l_1, l_2)$ irreduzibel. Sie sind aber grundreduzierbar.

Beachte

Ist t reduzierbar, so ist t auch grundreduzierbar.

Bemerkung

Wir benötigen den Begriff „grundreduzierbar" für den Test, ob $\text{GIrr}(R) = \text{GIrr}(R')$ aus Lemma 3.3 gilt.

Ein Lemma zur Grundreduzierbarkeit

Lemma 3.4

Sei R terminierend und C_R die durch R definierte Konstruktormenge. Enthält $t \in \mathsf{Term}(F, V)$ ein $f \in F - C_R$, so ist t grundreduzierbar.

Beweis: Übung.

Beispiel

Sei R das Regelsystem für $+$, app wie oben und

$F = \{0, s, +, ., \mathsf{nil}, \mathsf{app}\}$.

Es ist $C_R = \{0, s, \mathsf{nil}, .\}$.

Also sind $t_1 \equiv x + y$ und $t_2 \equiv \mathsf{app}(l_1, l_2)$ grundreduzierbar.

Komplexität der Grundreduzierbarkeit

Problem

Eingabe: R über sig $= (S, F, \alpha)$ und $t \in \mathsf{Term}(F, V)$.
Frage: Ist t in R grundreduzierbar?

Resultate

1. Für unbedingte R ist das Problem entscheidbar. Es ist schwer im allgemeinen Fall: Zeit $O(2^{2^n})$, falls $|t| = n$. Es ist viel leichter, falls R links-linear ist.
2. Für bedingte R ist das Problem unentscheidbar.

Beachte

a) Lemma 3.4 liefert in der Praxis häufig ein hinreichend starkes Kriterium für die Grundreduzierbarkeit und für den Test, ob $\mathsf{GIrr}(R_1) = \mathsf{GIrr}(R_2)$ gilt.

b) Wir geben noch ein Kriterium an, das noch etwas schärfer ist.

Test auf **GIrr**(R_1) = **GIrr**(R_2)

Seien

R_1, R_2 Regelsysteme über sig $= (S, F, \alpha)$.

Lemma 3.5

Für jede Regel $l \longrightarrow r$ in R_2 sei l in R_1 grundreduzierbar.

Dann gilt GIrr$(R_1) \subseteq$ GIrr(R_2).

Gilt zusätzlich $R_1 \subseteq R_2$, so gilt GIrr$(R_1) =$ GIrr(R_2).

Beweis:

Sei $t \in$ GIrr(R_1). Annahme: $t \notin$ GIrr(R_2).

t ist mit $l \longrightarrow r$ aus R_2 reduzierbar.

Es gibt σ mit $t \geq_{TT} \sigma(l)$.

t ist in R_1 reduzierbar, da $\sigma(l)$ in R_1 reduzierbar. $\lightning$

Sei zusätzlich $R_1 \subseteq R_2$.

Dann gilt GIrr$(R_2) \subseteq$ GIrr(R_1). Also gilt GIrr$(R_1) =$ GIrr(R_2).

Das Hauptergebnis

Sei spec $=$ (sig, R)
sig $= (S, F, \alpha)$
G Menge von sig-Gleichungen

Satz 3.2.

Seien R und R' grundkonvergent, R' ein Regelsystem zu $R \cup G$ und $R \subseteq R'$. Für jedes $l \longrightarrow r$ in $R' - R$ sei l in R grundreduzierbar.

Dann gilt $G \subseteq ITh$(spec).

Beweis: Dies folgt aus den letzten Lemmata.

Beachte

Es reicht, dass R und R' terminierend und grundkonfluent sind, man benötigt nicht die Konfluenz.

Literatur

Jouannaud + Kounalis, LICS 1986

Beispiel: Listen (1)

Sei

$F = \{\mathsf{nil}, ., \mathsf{app}, \mathsf{rev}\}$, $C = \{\mathsf{nil}, .\}$.

$$
\begin{array}{lll}
R: & \mathsf{app}(\mathsf{nil}, l_2) \longrightarrow l_2 & \\
 & \mathsf{app}(x.l_1, l_2) \longrightarrow x.\mathsf{app}(l_1, l_2) & \\
 & \mathsf{rev}(\mathsf{nil}) \longrightarrow \mathsf{nil} & \\
 & \mathsf{rev}(x.l) \longrightarrow \mathsf{app}(\mathsf{rev}(l), x.\mathsf{nil}) &
\end{array}
$$

$$
\begin{array}{llr}
G: & \mathsf{app}(\mathsf{app}(l_1, l_2), l_3) = \mathsf{app}(l_1, \mathsf{app}(l_2, l_3)) & (1) \\
 & \mathsf{rev}(\mathsf{rev}(l)) = l & (2) \\
 & \mathsf{rev}(\mathsf{app}(l_1, l_2)) = \mathsf{app}(\mathsf{rev}(l_2), \mathsf{rev}(l_1)) & (3)
\end{array}
$$

Es ist $C = C_R$ eine Konstruktormenge.

Vervollständigung von R und (2) liefert $R \cup R_1$ mit

$$
\begin{array}{ll}
R_1: & \mathsf{rev}(\mathsf{rev}(l)) \longrightarrow l \\
 & \mathsf{rev}(\mathsf{app}(l, x.\mathsf{nil})) \longrightarrow x.\mathsf{rev}(l)
\end{array}
$$

Also

$R_1 \subseteq ITh(\mathsf{spec})$.

Beispiel: Listen (2)

Vervollständigung von R und (1) liefert $R \cup R_2$ mit

$R_2 : \mathsf{app}(\mathsf{app}(l_1, l_2), l_3) \longrightarrow \mathsf{app}(l_1, \mathsf{app}(l_2, l_3))$

Also

(1) $\in ITh(\mathsf{spec})$.

Vervollständigung von $R \cup \{(1), (2)\}$ und von $R \cup \{(1), (3)\}$ divergiert.

Also

Vorsicht bei simultanen Beweisen mehrerer Hypothesen.

Vervollständigung von $R \cup G$ terminiert und liefert

$R = R_1 \cup R_2 \cup R_3$ mit

$R_3 : \mathsf{rev}(\mathsf{app}(l_1, l_2)) \longrightarrow \mathsf{app}(\mathsf{rev}(l_2), \mathsf{rev}(l_1))$.

Also

$G \subseteq ITh(\mathsf{spec})$.

Simultanes Beweisen mehrerer Hypothesen kann hilfreich sein.

Beispiel: Ganze Zahlen

Besonderheit

Nicht-freie Konstruktoren.

Sei

$F = \{0, s, p, +, -\},\ C = \{0, s, p\}$.

$$\begin{array}{lll}
R\!: & x + 0 \to x & x - 0 \to x \\
 & x + s(y) \to s(x + y) & x - s(y) \to p(x - y) \\
 & s + p(y) \to p(x + y) & x - p(y) \to s(x - y) \\
 & s(p(x)) \to x & p(s(x)) \to x
\end{array}$$

Es ist $C = C_R$ eine Konstruktormenge, aber nicht frei.

$$\begin{array}{lll}
G\!: & s(x) + y = s(x + y) & s(x) - y = s(x - y) \\
 & p(x) + y = p(x + y) & p(x) - y = p(x - y) \\
 & 0 + y = y & x - x = 0 \\
 & (x + y) - y = x & (x - y) + y = x
\end{array}$$

Vervollständigung von $R \cup G$ liefert $R' = R \cup R_1$ mit $R_1 = \vec{G}$.

Also

$G \subseteq ITh(\text{spec})$.

Beispiel: Natürliche Zahlen

Besonderheit: AC-Symbole.

Sei

$F = \{0, s, +\},\ C = \{0, s\}$.

$$\begin{array}{lll}
R\!: & x + 0 \to x & x + s(y) \to s(x + y) \\
 & & \\
G\!: & 0 + x = x & s(x) + y = s(x + y) \\
 & x + y = y + x & (x + y) + z = x + (y + z)
\end{array}$$

Es ist $C = C_R$ eine Konstruktormenge.

Vervollständigung von $R \cup G$ liefert $R' = R \cup R_1 \cup E^{\succ}$ mit

$$\begin{array}{lll}
R_1\!: & 0 + y \to y & s(x) + y \to s(x + y) \\
 & & (x + y) + z \to x + (y + z) \\
 & & \\
E\!: & x + y = y + x & x + (y + z) = y + (x + z)
\end{array}$$

$E^{\succ} = \{\sigma(u) \to \sigma(v) \mid u \doteq v$ in $E, \sigma(u) \succ \sigma(v)\}$

$\succ$ die LPO zu $+ \rhd s \rhd 0$.

Die linken Seiten der Regeln in $R_1 \cup E^{\succ}$ sind in R grundreduzierbar. R und R' sind grundkonvergent.

Also

$G \subseteq ITh(\text{spec})$.

Ein sparsames Verfahren: Motivation

Sei spec $=$ (sig, R) sig $= (S, F, \alpha)$
R unbedingtes Regelsystem
R grundkonfluent
$R \subseteq \succ$, $\succ$ eine Reduktionsordnung, total auf Term(F)
$G \subseteq$ Term$(F, V) \times$ Term(F, V)

Aufgabe: Überprüfe, ob $G \subseteq ITh$(spec) gilt.

Ziel

a) Die Vervollständigung von $R \cup G$ vermeiden: Sie kann divergieren oder gar abbrechen.
b) Weniger Inferenzschritte machen.

Ansatz

Die möglichen Inkonsistenzen verkleinern bis

i) es klar ist, dass keine Inkonsistenz existiert oder
ii) eine Inkonsistenz offensichtlich wird.

Literatur

Bachmair, LICS 1988

Nachweisbare Inkonsistenzen

Erinnerung

$s = t$ heißt konsistent (mit R), falls für alle $\sigma \in$ GSub$(s = t)$ gilt $\sigma(s) =_R \sigma(t)$, also auch $\sigma(s) \downarrow_R \sigma(t)$.

Definition 3.8

Die Gleichung $s = t$ heißt **nachweisbar inkonsistent** mit R, falls i), ii) oder iii) gilt

i) $s \succ t$ und s ist in R nicht grundreduzierbar.
ii) $t \succ s$ und t ist in R nicht grundreduzierbar.
iii) s, t sind $\succ$-unvergleichbar und $\#(s, t)$ ist nicht grundreduzierbar in $R \cup \{\#(x, x) \to x\}$. Dabei ist $\# \notin F$ neues Funktionssymbol.

G heißt **nachweisbar inkonsistent**, falls ein $s = t$ aus G nachweisbar inkonsistent ist.

Grundlagen des Verfahrens (1)

Idee Verbinde Suche nach Lösungen (mit Narrowing) und Suche nach Nicht-Lösungen (also Inkonsistenzen)

Ausführung Sei H Menge von Hypothesen

- Saturieren von H mit Narrowing:
  $s = t \in H$ simplifizieren.
  $s = t$ löschen, falls $s = t$ „offensichtlich konsistent“, sonst ersetzen durch $\mathrm{CP}(R, s = t)$.
  (Dies verkleinert Inkonsistenzen!)
- Stop, falls
  - $H = \emptyset \rightsquigarrow$ Antwort: JA
  - $s = t \in H$ ist „offensichtlich inkonsistent“ $\rightsquigarrow$ Antwort: NEIN

Ergebnis Verfahren mit Start $H = \{s = t\}$ liefert NEIN gdw $s = t \notin ITh(\text{spec})$

ohne Beweis

Grundlagen des Verfahrens (2)

Fragen

a) Wann ist $s = t$ „offensichtlich konsistent“?
b) Wann ist $s = t$ „offensichtlich inkonsistent“?

Beachte (Intuition)

a) $s = t$ ist „offensichtlich konsistent“ $\rightsquigarrow$ $s = t$ ist konsistent
   $\rightsquigarrow$ $s = t$ kann aus H gestrichen werden
b) $s = t$ ist „offensichtlich inkonsistent“
   $\rightsquigarrow$ $s = t$ ist inkonsistent $\rightsquigarrow$ Es gilt nicht $H \subseteq ITh(\text{spec})$

Antworten
zu a) Es gilt $s =_{I(E)} t$ (z.B. $s =_E t$)

zu b) viele Kriterien möglich, z.B.

b1) $s = t$ ist nachweisbar inkonsistent (Test ist aber schwer)
b2) $\mathrm{top}(s) = f \neq g = \mathrm{top}(t)$, $f, g \in C$ und C ist Menge von freien Konstruktoren

4 Positiv/negativ bedingte Spezifikationen

4.1 Motivation

Bisher (z. B. bei dem Cover-Set-Ansatz)

spec ist eine Spezifikation mit Konstruktoren in C, so dass

a) Jedes $f \in F - C$ ist total definiert.

b) Jede definierende Gleichung ist nur positiv bedingt.

Jetzt

Aufgabe der Voraussetzungen a) und b). Als Beweisziele sind beliebige (all-quantifizierten) Klauseln zugelassen.

Beachte

Zu a): Partialität ist schwer zu modellieren. Es gibt dazu verschiedene Ansätze. Unser Ansatz ist mit „totalen Algebren $\mathcal{A}$ mit Fehlerelementen". Die $f \in F - C$ werden zu partiellen Funktionen auf dem C-Redukt $\mathcal{A}^C$ von $\mathcal{A}$ interpretiert.

Zu b): Eine definierende Gleichung ist im Sinn der Logik eine Klausel mit ausgezeichnetem Kopf-Atom: $u = v \wedge u' \neq v' \Rightarrow s = t$. Wir werten sie so aus, dass wir „im wesentlichen" wieder in der Horn-Logik sind. Es gibt dann in den „Datenmodellen" initiale Modelle von spec.

Beispiel zur Partialität (1)

Sei $F = \{0, s, +, -\}$, $C = \{0, s\}$ und

$$E: \quad \begin{array}{ll} x + 0 = x & x - 0 = x \\ x + s(y) = s(x + y) & s(x) - s(y) = x - y \end{array}$$

Es ist $0 - s^{i+1}(0)$ nicht E-gleich zu einem C-Term, also ist $0 - s^{i+1}(0)$ ein Fehlerelement. Allgemein ist ein Fehlerelement ein $t \in \mathrm{Term}(F)$, der ein $0 - s^{i+1}(0)$ als Teilterm enthält. E-gleiche Grundterme werden identifiziert:

Sei $[t]$ die E-Äquivalenzklasse von $t \in \mathrm{Term}(F)$ und
$\mathcal{A} = (A, (f^{\mathcal{A}})_{f \in F})$ mit $A = \{[t] \mid t \in \mathrm{Term}(F)\}$ und

$$\begin{array}{ll} 0^{\mathcal{A}} = [0] & [t_1] +^{\mathcal{A}} [t_2] = [t_1 + t_2] \\ s^{\mathcal{A}}(t) = [s(t)] & [t_1] -^{\mathcal{A}} [t_2] = [t_1 - t_2] \end{array}$$

Dann ist das erweiterte C-Redukt von $\mathcal{A}$ die partielle Algebra

$$\mathcal{A}_e^C = (A^C, (f^{\mathcal{A}_e^C})_{f \in F}) \qquad A^C = \{[t] \mid t \in \mathrm{Term}(C)\}$$

Es ist z. B.

$$[s^i(0)] +^{\mathcal{A}^C{}_e} [s^j(0)] = [s^{i+j}(0)]$$

$$[s^i(0)] -^{\mathcal{A}^C{}_e} [s^j(0)] = \begin{cases} [s^{i-j}(0)] & i \geq j \\ \text{nicht definiert} & \text{sonst} \end{cases}$$

Offenbar ist $\mathcal{A}_e^C$ isomorph zu $\mathbb{N}$ mit der partiellen Differenz als $-^{\mathbb{N}}$.

Beispiel zur Partialität (2)

Beachte

spec kann man erweitern, damit $-$ „mehr definiert“ ist.

(1) $0 - y = 0$ oder (2) $0 - y = y$

Bei der Erweiterung (1) wird $\mathcal{A}^C$ bis auf Isomorphie zu $(\mathbb{N}, \{0, +1, +, \dot{-}\})$.

Bei der Erweiterung (2) wird $\mathcal{A}^C$ bis auf Isomorphie zu $(\mathbb{N}, \{0, +1, +, d)$ mit $d(x, y) = |x - y|$.

In allen 3 Fällen ist $\mathcal{A}$ ein **Datenmodell** zu spec.

Frage: Was gilt in allen Datenmodellen von spec?

Ziel

Definition der Gültigkeit einer Klausel C in spec, so dass die Gültigkeit unter Erweiterungen von spec erhalten bleibt.

Beispiel negative Bedingungen

Ansatz

Die Gleichheit wird in $\mathcal{A}$ klassisch ausgewertet,
d. h. $\mathcal{A}$ erfüllt $t_1 = t_2 \text{ gdw } t_1^{\mathcal{A}} = t_2^{\mathcal{A}}$.

Die Ungleichheit wird „konstruktiv“ ausgewertet,
d. h. $\mathcal{A}$ erfüllt $t_1 \neq t_2 \text{ gdw } t_1^{\mathcal{A}}, t_2^{\mathcal{A}} \in A^C$ und $t_1^{\mathcal{A}} \neq t_2^{\mathcal{A}}$.

Sei $F = \{0, s, \geq, d, \mathsf{true}, \mathsf{false}\}$, $C = \{0, s, \mathsf{true}, \mathsf{false}\}$.

$$
\begin{array}{lll}
E: & x \geq 0 = \mathsf{true} & d(x,y) = x \Leftarrow x \geq y = \mathsf{true} \\
 & s(x) \geq s(y) = x \geq y & d(x,y) = y \Leftarrow x \geq y \neq \mathsf{true}
\end{array}
$$

Sei $\mathcal{A}$ ein Modell von spec.

Es gilt $d(s^i(0), s^j(0)) = s^i(0)$ in $\mathcal{A}$ für $i \geq j$.

Für $i < j$ ist $s^i(0) \geq s^j(0)$ nicht notwendigerweise in $\mathcal{A}^C$, also muss in $\mathcal{A}$ nicht $d(s^i(0), s^j(0)) = s^j(0)$ gelten.

Zur E-Gleichheit:
Es gilt $s^i(0) \geq s^j(0) =_E \mathsf{true}$, also $d(s^i(0), s^j(0)) =_E s^i(0)$ für $i \geq j$.
Für $i < j$ gibt es kein $t \in \mathrm{Term}(C)$ mit $s^i(0) \geq s^j(0) =_E t$, also gilt nicht $s^i(0) \geq s^j(0) \neq_E \mathsf{true}$, also gilt nicht $d(s^i(0), s^j(0)) =_E s^j(0)$.

Zwei Typen von Variablen

Ansatz

Wir unterscheiden Konstruktorvariablen und generelle Variable:
Die Konstruktorvariablen $x \in V^C$ können in $\mathcal{A}$ nur Werte aus A^C annehmen.

Die generellen Variablen $x \in V^G$ können in $\mathcal{A}$ alle $a \in A$ als Wert annehmen, also auch „Fehlerelemente“.

Beachte

In den bisherigen Beispielen waren alle Variablen Konstruktorvariablen. Also durfte man in einer Gleichung z. B. nicht $0 - s(0)$ oder auch $s(0 - s^2(0))$ für x einsetzen.

Beispiel $F = \{0, s, -, \mathsf{nil}, ., \mathsf{app}\}$, $C = \{0, s, \mathsf{nil}, .\}$.

Sei $x, y \in V^C$ und $X, L_1, L_2 \in V^G$.

$$
\begin{array}{lll}
E: & x - 0 = x & s(x) - s(y) = x - y \\
 & \mathsf{app}(\mathsf{nil}, L_2) = L_2 & \mathsf{app}(X.L_1, L_2) = X.\mathsf{app}(L_1, L_2)
\end{array}
$$

Dies definiert die Konkatenation von Listen auch für den Fall, dass sie Fehlerelemente der Form $0 - s^i(0)$, $i > 0$, als Elemente haben.

4.2 Syntax und Semantik von Spezifikationen

Sei $\mathsf{sig} = (S, F, \alpha)$, $C \subseteq F$, $\mathsf{sig}^C = (S, C, \alpha|_C)$.

Dann heißt sig eine **Signatur über den Konstruktoren** in C.

Wir schreiben auch $\mathsf{sig} = (S, C, F, \alpha)$.

Sei $V = V^C \cup V^G$ mit $V^C = (V_s^C)_{s \in S}$ und $V^G = (V_s^G)_{s \in S}$.

Dann ist V ein **Variablensystem** zu $\mathsf{sig} = (S, C, F, \alpha)$.

Voraussetzung: $\mathsf{Term}_s(C) \neq \varnothing$ für alle $s \in S$.

Bezeichnung

Eine Substitution ist $\sigma : V \longrightarrow \mathsf{Term}(F, V)$.

σ ist eine **Konstruktorsubstitution**, falls $\sigma(x) \in \mathsf{Term}(C, V^C)$ für alle $x \in V^C$.

σ ist eine **induktive Substitution**, falls $\sigma(x) \in \mathsf{Term}(C)$ für $x \in V^C$ und $\sigma(x) \in \mathsf{Term}(F, V^G)$ für $x \in V^G$.

def ist ein Prädikatssymbol mit der festen Interpretation:

$\mathrm{def}(t)$ gdw t ist definiert.

Ein Atom A hat also die Form $\mathrm{def}(t)$ oder $t_1 = t_2$.

Es ist $\bar{l}$ die Negation eines Literals l, also $\overline{A} \equiv \neg A$, $\overline{\neg A} \equiv A$.

Spezifikationen

Sei sig $= (S, C, F, \alpha)$ und $V = V^C \cup V^G$.

Bezeichnung

Eine **bedingte Gleichung** ist eine Klausel C mit einem ausgezeichneten Atom $t_1 = t_2 \in C$, also $C \equiv t_1 = t_2 \vee l_1 \vee \cdots \vee l_n$.

Schreibweise: $t_1 = t_2 \Leftarrow \overline{l_1} \wedge \overline{l_2} \wedge \cdots \wedge \overline{l_n}$.

Definition 4.1

Eine **Basisspezifikation** ist spec^C = (sig^C, E^C) über V^C, wobei die Gleichungen aus E^C nur positiv bedingt sind.

Eine **Spezifikation** über spec^C ist spec = (sig, E) über V mit $E = E^C \cup E^F$, wobei E^F aus bedingten Gleichungen besteht.

Beachte zur Intention:

Die Elemente aus C heißen Konstruktoren. E^C definiert den Zusammenhang zwischen den Konstruktoren. Sie heißen **frei**, falls E^C leer ist. (Dies ist in Anwendungen häufig der Fall.) E^F definiert (partielle) Funktionen auf den Konstuktortermen.

sig-Algebren

Sei sig $= (S, C, F, \alpha)$.

Definition 4.2

Eine **sig-Algebra** $\mathcal{A} = ((A_s)_{s\in S}, (f^{\mathcal{A}})_{f \in F})$ ist wie üblich definiert. Sei $t^{\mathcal{A}}$ die Auswertung von $t \in$ Term(F) in $\mathcal{A}$.
Das C-**Redukt** $\mathcal{A}^C$ von $\mathcal{A}$ ist die sig^C-Algebra
$\mathcal{A}^C = ((A_s^C)_{s\in S}, (f^{\mathcal{A}^C})_{f\in C})$ mit

$A_s^C = \{t^{\mathcal{A}} \mid t \in \text{Term}_s(C)\}$ für alle $s \in S$

$f^{\mathcal{A}^C}(a_1, \ldots, a_n) = f^{\mathcal{A}}(a_1, \ldots, a_n)$ für alle $f \in C$

Beachte

Gilt $a_1, \ldots, a_n \in A^C$, so gibt es $t_i \in$ Term(C) mit $a_i = t_i^{\mathcal{A}}$.

Ist $f \in C$, so ist $f(t_1, \ldots, t_n) \in$ Term(C) und $f^{\mathcal{A}^C}(a_1, \ldots, a_n) = f(t_1, \ldots, t_n)^{\mathcal{A}} \in A^C$. Also sind $\mathcal{A}$ und $\mathcal{A}^C$ „totale Algebren“, d. h., alle $f^{\mathcal{A}}$ mit $f \in F$ bzw. $f^{\mathcal{A}^C}$ mit $f \in C$ sind totale Funktionen.

Lemma 4.1

Seien $\mathcal{A}$ und $\mathcal{B}$ sig-Algebren und $\varphi = (\varphi_s)_{s\in S}$ ein sig-Homomorphismus $\varphi : \mathcal{A} \to \mathcal{B}$. Sei $\varphi^C = (\varphi_s^C)_{s\in S}$ die Einschränkung von φ auf $\mathcal{A}^C$. Dann ist φ^C ein sig^C-Homomorphismus $\varphi^C : \mathcal{A}^C \to \mathcal{B}^C$.

sig-Algebren (Forts.)

Die Funktionen zu $f \in F - C$ sind als partielle Funktionen über A^C erklärbar.

Definition 4.3

Das **erweiterte** C**-Redukt** $\mathcal{A}_e^C$ von $\mathcal{A}$ ist die sig-Algebra $\mathcal{A}_e^C = ((A_s^C)_{s\in S}, (f^{\mathcal{A}_e^C})_{f\in F})$ mit

$$f^{\mathcal{A}_e^C}(a_1,\ldots,a_n) = f^{\mathcal{A}^C}(a_1,\ldots,a_n) \qquad \text{für alle } f \in C$$

$$f^{\mathcal{A}_e^C}(a_1,\ldots,a_n) = \begin{cases} f^{\mathcal{A}}(a_1,\ldots,a_n) & \text{falls} \\ & f^{\mathcal{A}}(a_1,\ldots,a_n) \in A^C \\ \text{nicht definiert} & \text{sonst} \end{cases}$$

$$\text{für alle } f \in F - C$$

spec-Modelle

Sei spec $=$ (sig, E) eine Spezifikation über spec^C,
$\mathcal{A}$ eine sig-Algebra.

Definition 4.4

a) Eine **Variablen-Belegung** $\varphi = (\varphi^C, \varphi^G)$ in $\mathcal{A}$ ist eine Abbildung $\varphi : V \to A$ mit $\varphi^C : V^C \to A^C$ und $\varphi^G : V^G \to A$. Sie wird homomorph fortgesetzt zu φ : Term$(F, V) \to A$.

b) $(\mathcal{A}, \varphi)$ erfüllt $t_1 = t_2$, falls $\varphi(t_1) = \varphi(t_2)$ in $\mathcal{A}$. $(\mathcal{A}, \varphi)$ erfüllt $t_1 \neq t_2$, falls $\varphi(t_1) \neq \varphi(t_2)$ in $\mathcal{A}$. $(\mathcal{A}, \varphi)$ erfüllt $\mathrm{def}(t)$, falls $\varphi(t) \in A^C$. $(\mathcal{A}, \varphi)$ erfüllt eine Klausel D, falls $(\mathcal{A}, \varphi)$ ein Literal aus D erfüllt.

c) $\mathcal{A}$ **erfüllt** D (oder D gilt in $\mathcal{A}$), falls für jedes φ gilt: $(\mathcal{A}, \varphi)$ erfüllt D. $\mathcal{A}$ erfüllt E, falls $\mathcal{A}$ jede bedingte Gleichung aus E erfüllt. Dann heißt $\mathcal{A}$ ein **Modell** von spec.

Schreibweise

$\mathcal{A} \models E$, falls $\mathcal{A}$ E erfüllt.

$\mathcal{K} \models E$, falls $\mathcal{A} \models E$ gilt für alle $\mathcal{A} \in \mathcal{K}$.

Mod(spec) ist die Klasse aller Modelle von spec.

Bemerkungen

1. Bemerkung

Die Definition von „$\mathcal{A}$ erfüllt D" legt fest, dass def stets als das Definiertheitsprädikat interpretiert wird.

Sie berücksichtigt auch, dass alle Gleichungen in E implizit allquantifiziert sind.

2. Bemerkung

In Spezifikationen soll die Ungleichheit stets konstruktiv ausgewertet werden. Wir erreichen dies später dadurch, dass jede Ungleichung $t_1 \neq t_2$ durch $\mathrm{def}(t_1)$ und $\mathrm{def}(t_2)$ bewacht wird.

Gilt $t \in \mathsf{Term}(C, V^C)$, so gilt $\mathrm{def}(t)$ automatisch, kann also weggelassen werden.

3. Bemerkung

spec^C ist eine positiv-bedingte Spezifikation, also ist $I(\mathsf{spec}^C)$ ein Modell von spec^C.

Mod(spec) ist nicht leer, denn das **finale Modell** $\mathcal{A}_{\mathsf{fin}}$ liegt immer in Mod(spec). Dabei ist $\mathcal{A} = \mathcal{A}_{\mathsf{fin}} = ((A_s)_{s\in S}, (f^{\mathcal{A}})_{f\in F})$ die sig-Algebra, in der jedes $A_s = \{a_s\}$ nur aus einem Element besteht. Es gilt $f^{\mathcal{A}}(a_{s_1}, \ldots, a_{s_n}) = a_s$ für $f : s_1 \ldots s_n \to s$.

Datenmodelle

Sei spec eine Spezifikation über spec^C.

Intention

Eine sig-Algebra $\mathcal{A} \in$ Mod(spec) ist ein Datenmodell, wenn das C-Redukt $\mathcal{A}^C$ aus den „Daten" von $I(\mathsf{spec}^C)$ besteht, d. h., $\mathcal{A}^C$ isomorph zu $I(\mathsf{spec}^C)$ ist. Dazu müssen wir uns aber auf „zulässige" Spezifikationen beschränken. Siehe später.

Definition 4.5

$\mathcal{A} \in$ Mod(spec) heißt ein **Datenmodell**, falls für alle $t_1, t_2 \in \mathsf{Term}(C)$ gilt: $t_1^{\mathcal{A}} = t_2^{\mathcal{A}}$ impliziert Mod(spec) $\models t_1 = t_2$. DMod(spec) ist die Klasse der Datenmodelle von spec.

Lemma 4.2

Sei $\mathcal{A} \in$ Mod(spec) und $\mathcal{K}$ die Klasse der sig^C-Algebren $\mathcal{B}^C$ mit $\mathcal{B} \in$ Mod(spec). Dann gilt $\mathcal{A} \in$ DMod(spec) gdw $\mathcal{A}^C$ ist initial in $\mathcal{K}$.

Also: Gilt $\mathcal{A}, \mathcal{B} \in$ DMod(spec), so sind $\mathcal{A}^C$ und $\mathcal{B}^C$ sig^C-isomorph.

Zulässige Spezifikationen (1)

Intention

Wir benötigen zusätzliche Bedingungen an Spezifikationen, so dass

a) man in spec effektiv rechnen kann,
b) der Begriff „Datenmodell“ stets sinnvoll ist, z. B. DMod(spec) nicht leer ist.

Ansatz

a) E ist als (i. Allg. konvergentes) Regelsystem R gegeben.
b) spec ist „konservative“ Erweiterung von spec^C.

Definition 4.6

spec ist eine **konservative Erweiterung** von spec^C, falls für alle $t_1, t_2 \in \mathsf{Term}(C)$ gilt: $t_1 =_E t_2 \text{ gdw } t_1 =_{E^C} t_2$.

Beispiel $F = \{c_1, c_2, d\}$, $C = \{c_1, c_2\}$, $E^C = \varnothing$.

(1) $d = c_1$ (2) $d = c_2$

Besteht E nur aus (1), so ist spec $=$ (sig, E) eine konservative Erweiterung von spec^C. Dies gilt nicht, wenn E aus (1) und (2) besteht.

Zulässige Spezifikationen (2)

Sei sig $= (S, C, F, \alpha)$.

Definition 4.7

Eine **Konstruktorregel** ist $l \to r \Leftarrow u = v$ mit

- $l, r, u_i, v_i \in \mathsf{Term}(C, V^C)$.
- $\mathsf{Var}(r) \subseteq \mathsf{Var}(l)$ und $\mathsf{Var}(u_i, v_i) \subseteq \mathsf{Var}(l)$.

Eine **definierende Regel** ist $l \to r \Leftarrow \Delta$ mit

- $l \in \mathsf{Term}(F, V) - \mathsf{Term}(C, V)$.
- Δ ist Konjunktion von Literalen nicht von der Form $\neg\mathrm{def}(t)$.

Also

Eine Konstruktorregel ist nur positiv bedingt.

Eine definierende Regel ist positiv/negativ bedingt. Es gibt keine Bedingung der Form $\neg\mathrm{def}(t)$. Die linke Seite l enthält ein Funktionssymbol $f \in F - C$.

Zulässige Spezifikationen (3)

Sei sig $= (S, C, F, \alpha)$.

Definition 4.8

Eine **zulässige Spezifikation** ist spec $=$ (sig, R) mit

a) $R = R^C \cup R^D$, wobei R^C eine Menge von Konstruktorregeln und R^D eine Menge von definierenden Regeln ist.

b) $\rightarrow_R$ ist konfluent (zur Definition von $\rightarrow_R$ siehe unten).

c) Für jede Regel $l \rightarrow r \Leftarrow \Delta$ gilt: Ist $t_1 \neq t_2$ in Δ, $i \in \{1,2\}$ und $t_i \notin$ Term(C, V^C), so ist $\mathrm{def}(t_i)$ in Δ.

Definition 4.9

Sei $R = R^C \cup R^D$. (Beachte: $t_1 \neq t_2$ steht für $\neg t_1 = t_2$).

a) Es ist $\rightarrow_{R^C}$ wie üblich erklärt:
$t_1 \rightarrow_{R^C} t_2$ gdw es gibt $l \rightarrow r \Leftarrow u = v$ in R_C und σ, so dass $t_1 \rightarrow t_2$ mit $l \rightarrow r$, σ und $\sigma(u) \downarrow_{R^C} \sigma(v)$.

b) Zur Definition $\rightarrow_R$:
σ erfüllt $t_1 = t_2$ gdw $\sigma(t_1) \downarrow_R \sigma(t_2)$.
σ erfüllt $\mathrm{def}(t)$ gdw $\sigma(t) \xrightarrow{*}_R t_0$ mit $t_0 \in$ Term(C, V^C).
σ erfüllt Δ gdw σ erfüllt jedes Literal aus Δ.
$t_1 \rightarrow_R t_2$ gdw i) $t_1 \rightarrow_{R^C} t_2$ oder
ii) es gibt $l \rightarrow r \Leftarrow \Delta$ in R^D mit $t_1 \rightarrow t_2$ mit $l \rightarrow r, \sigma$ und σ erfüllt Δ.

Bemerkungen

1. Bemerkung

$\rightarrow_R$ wird in zwei Schichten definiert:

1. Schicht: $\rightarrow_{R^C}$ benutzt nur R^C-Regeln.

2. Schicht: $\rightarrow_R$ benutzt alle R-Regeln.

2. Bemerkung

Die Definitionen von $\rightarrow_{R^C}$ und $\rightarrow_R$ sind rekursiv.

Also: $\rightarrow_{R^C}$ und $\rightarrow_R$ sind die kleinsten Relationen, die die Bedingungen der Definitionen erfüllen.

3. Bemerkung

Ist $l \rightarrow r \Leftarrow \Delta$ in R^D, so enthält l ein $f \in F - C$. Die Regeln aus R^D sind also nicht auf Terme $t \in$ Term(C, V^C) anwendbar.

Also gilt in einer zulässigen Spezifikation für alle $t_1, t_2 \in$ Term(C, V^C):

$$t_1 =_R t_2 \;\text{ gdw }\; t_1 \downarrow_R t_2 \;\text{ gdw }\; t_1 \downarrow_{R^C} t_2 \;\text{ gdw }\; t_1 =_{R^C} t_2$$

Lemma 4.3

Ist spec eine zulässige Spezifikation, so ist spec eine konservative Erweiterung zu spec^C.

Beispiel

Sei $C = \{0, s, p\}$ $\quad D = \{f, \mathsf{half}\}$ $\quad F = C \cup D$
$R = R^C \cup R^D$

R^C: $s(p(x)) \to x \quad p(s(x)) \to x$

R^D: $\mathsf{half}(0) \to 0$
$\mathsf{half}(s(s(x))) \to s(\mathsf{half}(x))$
$\mathsf{half}(p(p(x))) \to p(\mathsf{half}(x))$
$s(\mathsf{half}(p(x))) \to \mathsf{half}(s(x))$
$p(\mathsf{half}(s(x))) \to \mathsf{half}(p(x))$
$f(x) \to 0 \Leftarrow \mathsf{half}(x) \neq x, \mathrm{def}(\mathsf{half}(x))$

Dann gilt:

a) Die Spezifikation ist zulässig.

b) Jedes Modell $\mathcal{A} = (\mathbb{Z}, F^{\mathcal{A}})$ aus Mod(spec) mit

$$0^{\mathcal{A}} = 0 \qquad s^{\mathcal{A}}(x) = x + 1 \qquad p^{\mathcal{A}}(x) = x - 1$$

ist ein Datenmodell. Beispiele dazu sind

$\mathsf{half}^{\mathcal{A}}(2x) = x \quad \mathsf{half}^{\mathcal{A}}(2x+1) = x + 1$
$f^{\mathcal{A}}(x) = 0$ und
$\mathsf{half}^{\mathcal{A}}(2x) = x \quad \mathsf{half}^{\mathcal{A}}(2x+1) =\uparrow$
$f^{\mathcal{A}}(2x) = 0 \quad f^{\mathcal{A}}(2x+1) =\uparrow$

c) Die Algebra $\mathcal{A} = (\mathbb{Z}, F^{\mathcal{A}})$ mit

$$0^{\mathcal{A}} = 0 \qquad s^{\mathcal{A}}(x) = p^{\mathcal{A}}(x) = \mathsf{half}^{\mathcal{A}}(x) = f^{\mathcal{A}}(x) = x$$

ist ein Modell von spec, aber kein Datenmodell. Es gilt z. B. $\mathcal{A} \models s(0) = 0$, aber nicht Mod(spec) $\models s(0) = 0$.

Das Standard-Datenmodell

Sei sig $= (S, C, F, \alpha)$,
spec $=$ (sig, R) eine zulässige Spezifikation.

Definition 4.10

Das **Standard-Datenmodell** zu spec ist die sig-Algebra $\mathcal{M} = \mathcal{M}(\text{spec})$ mit

$$M_s = \{[t] \mid t \in \mathrm{Term}(F)\} \qquad \text{für } s \in S$$
$$f^{\mathcal{M}}([t_1], \ldots, [t_n]) = [f(t_1, \ldots, t_n)] \qquad \text{für } f \in F$$

Dabei ist $[t] = \{t' \mid t =_R t'\}$ die R-Äquivalenzklasse zu t.

Beachte

Ist R konvergent, so ist $\mathcal{M}$(spec) isomorph zur kanonischen Termalgebra mit den R-irreduziblen Grundtermen als Elementen.

Lemma 4.4

a) $\mathcal{M} = \mathcal{M}(\text{spec})$ ist ein Datenmodell.

b) Ist $\mathcal{A} \in$ DMod(spec), so sind $\mathcal{A}^C$ und $\mathcal{M}^C$ isomorph.

c) $\mathcal{M}^C$ hat die R^C-Äquivalenzklassen der $t \in \mathrm{Term}(C)$ als Elemente.

Bemerkung zum Standard-Datenmodell

Sei spec $=$ (sig, R) zulässige Spezifikation über
sig $= (S, C, F, \alpha)$ und $V = V^C$.

1. Bemerkung

Sei spec eine positiv-bedingte Spezifikation und seien alle $f \in F - C$ total definiert (d. h., für alle $t \in \mathsf{Term}(F)$ gibt es $t' \in \mathsf{Term}(C)$ mit $t =_R t'$).
Dann ist in $\mathcal{M}_e^C$ jedes $f^{\mathcal{M}_e^C}$ total definiert und $\mathcal{M}_e^C = \mathcal{M}$ ist isomorph zum initialen Modell $\mathcal{I} = \mathcal{I}(\mathsf{spec})$ von früher.

2. Bemerkung Sei $t, t_1, t_2 \in \mathsf{Term}(F)$

Gilt $\mathcal{M} \models \mathrm{def}(t)$, so gilt $\mathcal{A} \models \mathrm{def}(t)$ für alle $\mathcal{A} \in \mathsf{DMod}(\mathsf{spec})$.
Gilt $\mathcal{M} \not\models \mathrm{def}(t)$, so kann $\mathcal{A} \models \mathrm{def}(t)$ für ein $\mathcal{A} \in \mathsf{DMod}(\mathsf{spec})$ gelten.
Gilt $\mathcal{M} \models t_1 = t_2$, so gilt $\mathcal{A} \models t_1 = t_2$ für alle $\mathcal{A} \in \mathsf{DMod}(\mathsf{spec})$.
Gilt $\mathcal{M} \not\models t_1 = t_2$, so kann $\mathcal{A} \models t_1 = t_2$ für ein $\mathcal{A} \in \mathsf{DMod}(\mathsf{spec})$ gelten.
Aber: Gilt $\mathcal{M} \models \mathrm{def}(t_1) \wedge \mathrm{def}(t_2)$, so gilt $\mathcal{M} \models t_1 = t_2$ gdw $\mathsf{DMod}(\mathsf{spec}) \models t_1 = t_2$.

3. Bemerkung

Ziel ist es, für Klauseln D zu zeigen $\mathsf{DMod}(\mathsf{spec}) \models D$. Die 2. Bemerkung zeigt, dass es reicht $\mathcal{M} \models D$ zu zeigen, falls $\mathcal{M} \models \mathrm{def}(t_1) \wedge \mathrm{def}(t_2)$ gilt für alle negativen Literale $t_1 \neq t_2$ in D. Also: $\mathcal{M} = \mathcal{M}(\mathsf{spec})$ spielt eine ähnliche Rolle wie früher $\mathcal{I} = \mathcal{I}(\mathsf{spec})$.

Ergebnisse zum Standard-Datenmodell

Sei sig $= (S, C, F, \alpha)$, spec $=$ (sig, R) zulässig,
$\mathcal{M} = \mathcal{M}(\mathsf{spec})$.

Lemma 4.5

Für alle $t_1, t_2 \in \mathsf{Term}(F, V^G)$ gilt

$$t_1 =_R t_2 \ \mathrm{gdw}\ \mathsf{DMod}(\mathsf{spec}) \models t_1 = t_2.$$

Satz 4.1.

Sei D eine Klausel und für alle Literale $t_1 \neq t_2$ in D gelte $\mathcal{M} \models \mathrm{def}(t_i)$ für $i = 1, 2$. Dann gilt: Aus $\mathcal{M} \models D$ folgt $\mathsf{DMod} \models D$.

Bemerkung Sei D eine Klausel.

a) Sei $D \equiv A_1 \wedge \cdots \wedge A_m \Rightarrow B_1 \vee \cdots \vee B_n$ und die A_i, B_j Gleichheitsatome. Der Satz besagt:
Gilt $\mathcal{M} \models \mathrm{def}(t)$ für alle t, die in den A_i auftreten und gilt $\mathcal{M} \models D$, so gilt auch $\mathcal{A} \models D$ für alle Datenmodelle $\mathcal{A}$.
b) Wir entwickeln Hilfsmittel zum Nachweis von $\mathsf{DMod}(\mathsf{spec}) \models D$.

Konfluenzkriterien (1)

Frage

Wie zeigt man, dass spec $= (\text{sig}, C, R)$ zulässig ist.

Also: Wie zeigt man die Konfluenz von R?

Beachte

Der Konfluenz-Nachweis für R ist i. Allg. schwer, besonders dann, wenn R nicht terminierend ist. Wir formulieren hinreichende Bedingungen mit „kritischen Paaren".

Definition 4.11

Sei $R = R^C \cup R^D$ wie stets.

Seien $\varrho_i \equiv l_i \to r_i \Leftarrow \Delta_i$, $i = 1, 2$, variablen-disjunkte Regeln aus R, sei $\pi \in \text{Pos}(l_2)$ mit $l_2/\pi \notin V$ und sei $\sigma = \text{mgu}(l_1, l_2/\pi)$. Ist $\sigma(l_2[r_1]_\pi) \not\equiv \sigma(r_2)$, so ist

$$\langle(\sigma(l_2[r_1]_\pi), \sigma(\Delta_1), a_1), (\sigma(r_2), \sigma(\Delta_2), a_2)\rangle$$

ein **kritisches Paar** vom Typ (a_1, a_2). Dabei gilt $a_i = 0$, falls $\varrho_i \in R^C$, und $a_i = 1$, falls $\varrho_i \in R^D$.

Das kritische Paar heißt **komplementär**, falls (i) oder (ii) gilt:

i) es gibt ein Atom $u = v$, das in einem $\sigma(\Delta_i)$ positiv und in dem anderen negativ vorkommt.

ii) es gibt t und irreduzible Grundterme u und v, so dass $t = u$ in $\sigma(\Delta_1)$ und $t = v$ in $\sigma(\Delta_2)$ vorkommt.

Konfluenzkriterien (2)

Beispiel

Seien true und false irreduzibel.

Jedes kritische Paar zu $l_1 \to r_1 \Leftarrow \Delta_1$ und $l_2 \to r_2 \Leftarrow \Delta_2$ ist komplementär, falls Δ_1 die Gleichung $u =$ true und Δ_2 entweder $u =$ false oder $u \neq$ true enthält.

Also: Fallunterscheidungen führen häufig zu kritischen Paaren, die komplementär sind.

Definition 4.12

R heißt **links-linear**, falls für jede Regel $l \to r \Leftarrow \Delta$ gilt, dass l keine Variable mehrfach enthält.

R heißt **schwach normal**, falls für jede Regel $l \to r \Leftarrow \Delta$ gilt:

Für jedes $t_1 = t_2$ in Δ gilt für ein $i \in \{1, 2\}$: t_i ist ein irreduzibler Grundterm oder $t_i \in \text{Term}(C, V^C)$ oder es ist $\text{def}(t_i)$ in Δ.

Definition 4.13

R heißt **orthogonal**, falls R linkslinear ist und keine kritischen Paare existieren.

Konfluenzkriterien (3)

Satz 4.2.

Jedes orthogonale R ist konfluent.

ohne Beweis.

Satz 4.3.

Sei spec $= (\text{sig}, C, R)$ eine Spezifikation über spec$^C = (\text{sig}^C, R^C)$.

Sei $R = R^C \cup R^D$ linkslinear und schwach normal.

Sei R^C konfluent und jedes kritische Paar vom Typ $(0,1)$, $(1,0)$ oder $(1,1)$ sei komplementär.

Dann ist R konfluent.

ohne Beweis.

Beachte

Hier wird nicht verlangt, dass R terminierend ist.

Ist R verkleinernd (decreasing), so gibt es schwächere Kriterien an die kritischen Paare, die hinreichend sind für die Konfluenz.

Induktive Gültigkeit

Ziel

Sinnvolle Definition von „Die Klausel D ist gültig in spec“.

2 Ansätze sind naheliegend

a) D gilt in allen Datenmodellen.

b) D gilt in $\mathcal{M}(\text{spec})$.

Wunsch: Monotonie-Eigenschaft

Gilt D in spec und ist spec$'$ eine „vernünftige“ Erweiterung von spec, so gilt D auch in spec$'$. Dies führt zu Ansatz a).

Dann

Man kann beim Aufbau der Spezifikation schritthaltend beweisen. Dabei bleiben bewiesene Aussagen gültig.

Definition 4.14

Sei spec eine zulässige Spezifikation und D eine Klausel. D heißt **gültig** oder ein **induktives Theorem** in spec, falls DMod(spec) $\models D$ gilt.

Konstruktorkonsistente Erweiterungen

Definition 4.15

Seien $\text{spec}_i = (\text{sig}_i, C_i, R_i)$ Spezifikation über $\text{sig}_i = (S_i, F_i, \alpha_i)$ und Variablensystemen V_i für $i = 0, 1$.

spec_1 ist eine **konstruktorkonsistente Erweiterung** von spec_0, falls gilt

(1) $S_0 \subseteq S_1$, $F_0 \subseteq F_1$, $C_0 \subseteq C_1$, $\alpha_0 \subseteq \alpha_1$.

(2) $V_{0,s} = V_{1,s}$ für alle $s \in S_0$.

(3) $R_0 \subseteq R_1$.

(4) Ist $c \in C_1 - C_0$ und $c : s_1 \dots s_n \to s$, so ist $s \notin S_0$.

(5) Ist $l \to r \Leftarrow u = v$ in $R_1^C - R_0^C$, so ist $l \notin \text{Term}(C_0, V_0)$.

Satz 4.4.

Sei spec_1 eine konstruktorkonsistente Erweiterung von spec_0 und seien $\text{spec}_1, \text{spec}_0$ zulässig. Sei D eine Klausel. Gilt $\text{DMod}(\text{spec}_0) \models D$, so gilt auch $\text{DMod}(\text{spec}_1) \models D$.

ohne Beweis.

Beispiel 1

Sei $F = \{0, s, +, -\}$, $C = \{0, s\}$.

$$
\begin{array}{llll}
R: & x + 0 \to x & R': & x + 0 \to x \\
 & x + s(y) \to s(x + y) & & x + s(y) \to s(x + y) \\
 & x - 0 \to x & & x - 0 \to x \\
 & s(x) - s(y) \to x - y & & s(x) - s(y) \to x - y \\
 & & & 0 - y \to 0
\end{array}
$$

Sei $\mathcal{M} = \mathcal{M}(\text{spec})$ und $\mathcal{M}' = \mathcal{M}(\text{spec}')$.

$D_1 \equiv (x + y) - y = x$

$D_2 \equiv 0 - s(0) \neq 0$.

Es gilt $\text{DMod}(\text{spec}) \models D_1$, also ist D_1 gültig in spec.

Es gilt $\mathcal{M} \models D_2$, aber $\mathcal{M}' \not\models D_2$. Da $\mathcal{M}'$ ein Datenmodell zu spec ist, gilt nicht $\text{DMod}(\text{spec}) \models D_2$.

Beachte

spec' ist eine konstruktorkonsistente Erweiterung von spec. Definierte man „gültig in spec“ als „gültig in $\mathcal{M}(\text{spec})$“, so bliebe die Gültigkeit nicht erhalten bei konstruktorkonsistenten Erweiterungen.

Beispiel 2

Sei spec wie in Beispiel 1.

Es gilt

DMod(spec) $\models x - y = 0 \wedge y - x = 0 \Rightarrow x = y$.

Beweis:

Sei $\mathcal{A} \in$ DMod(spec), $\sigma : \{x, y\} \to \mathsf{Term}(C)$.

Zu zeigen ist
$\mathcal{A} \models \sigma(x) - \sigma(y) = 0 \wedge \sigma(y) - \sigma(x) = 0 \Rightarrow \sigma(x) = \sigma(y)$.
Sei $\sigma(x) = s^i(0)$ und $\sigma(y) = s^j(0)$ und $i \geq j$.

Es gilt $\mathcal{M}$(spec) $\models s^i(0) - s^j(0) = s^{i-j}(0)$, also auch $\mathcal{A} \models s^i(0) - s^j(0) = s^{i-j}(0)$.

Wegen $\mathcal{A} \in$ DMod(spec) gilt $\mathcal{A} \models s^{i-j}(0) = 0 \text{ gdw } i = j$.

Für $i \leq j$ verläuft der Beweis analog mit $\sigma(y) - \sigma(x) = 0$.

Also: Aus $\mathcal{A} \models \sigma(x) - \sigma(y) = 0 \wedge \sigma(y) - \sigma(x) = 0$ folgt $\sigma(x) = \sigma(y)$.

Also
$\mathcal{A} \models \sigma(x) - \sigma(y) = 0 \wedge \sigma(y) - \sigma(x) = 0 \Rightarrow \sigma(x) = \sigma(y)$.

Also DMod(spec) $\models x - y = 0 \wedge y - x = 0 \Rightarrow x = y$.

Beispiel 3

Sei $F = \{0, s, \geq, \mathsf{true}, \mathsf{false}\}$, $C = \{0, s, \mathsf{true}, \mathsf{false}\}$,
$S = \{\mathsf{NAT}, \mathsf{BOOL}\}$,
$R : x \geq 0 \to \mathsf{true}$
$s(x) \geq s(y) \to x \geq y$

Es gilt nicht

DMod(spec) $\models x \geq y = \mathsf{true} \wedge y \geq x = \mathsf{true} \Rightarrow x = y$.

Beweis: Sei $\mathcal{A} = (\mathbb{N}, \mathbb{B}; F^{\mathcal{A}})$, $\mathbb{B} = \{T, F\}$ mit $0^{\mathcal{A}} = 0$, $s^{\mathcal{A}}(x) = x + 1$, $\mathsf{true}^{\mathcal{A}} = T$, $\mathsf{false}^{\mathcal{A}} = F$ und $x \geq^{\mathcal{A}} y = T$ für alle $x, y \in \mathbb{N}$

Dann gilt
$\mathcal{A} \in$ DMod(spec) und
$\mathcal{A} \not\models x \geq y = \mathsf{true} \wedge y \geq x = \mathsf{true} \Rightarrow x = y$,
also nicht DMod(spec) $\models x \geq y = \mathsf{true} \wedge y \geq x = \mathsf{true} \Rightarrow x = y$.

Beachte die Ähnlichkeit zu Beispiel 2.

Es gilt

$\mathcal{M}$(spec) $\models x \leq y = \mathsf{true} \wedge y \leq x = \mathsf{true} \Rightarrow x = y$.

4.3 Der Ansatz von QUODLIBET

Bemerkung

QUODLIBET ist ein in der AG entwickeltes Beweissystem zum Nachweis von induktiven Theoremen zu positiv/negativ bedingten Spezifikationen (mit nur partiell spezifizierten Funktionen).

QUODLIBET

- basiert auf einem Inferenzsystem,
- fasst mehrere Inferenzschritte zu Taktiken zusammen,
- erlaubt die Definition neuer Taktiken,
- ist im Moment nur hochgradig interaktiv einsetzbar.

QUODLIBET

- analysiert die Definitionen der $f \in F - C$,
- arbeitet nur mit freien Konstruktoren,
- erlaubt die Verwendung von Lemmata,
- schlägt sinnvolle Induktionsschemata vor,
- baut Beweisbäume auf,
- erlaubt die Manipulation von Beweisbäumen z. B. durch Aufruf von Taktiken, Ausführung von einzelnen Inferenzschritten, Rücksetzungen von Inferenzfolgen,...

Der prinzipielle Ansatz

QUODLIBET

- arbeitet auf Klauseln,
- verfolgt das Prinzip der Verkleinerung von Gegenbeispielen bis
  a) die Existenz eines Gegenbeispiels evident wird (dann war das Beweisziel kein Theorem) oder
  b) die Nicht-Existenz eines Gegenbeispiels evident wird (dann war das Beweisziel ein Theorem).

Regeln werden auf Klauseln angewandt und sind

- nicht-applikativ
  (z. B. Reinigen einer Klausel oder Aufspaltung nach Fällen) oder
- applikativ
  (z. B. Anwenden einer definierenden Gleichung, eines Lemmas oder einer Induktionsvoraussetzung auf die Klausel).

Verwendung einer Induktionsordnung

- sehr flexibel,
- wird erst spät „geeignet“ festgelegt,
- Nachweis von „kleiner“ zur Anwendung einer Induktionsvoraussetzung ist eine Beweisaufgabe.

Voraussetzungen/Bezeichnungen

Voraussetzungen generell ab jetzt

spec $=$ (sig, E) ist eine (zulässige) Spezifikation.

Die Konstruktoren sind frei, also $E^C = \varnothing$.

Es gibt keine generellen Variablen, also $V^G = \varnothing$ (der Einfachheit halber).

Bezeichnungen (gemäß U. Kühler, Dissertation, 2000)

λ bezeichnet ein Literal.

$\varGamma$ bezeichnet eine Klausel.

$\langle \varGamma; \omega \rangle$ ist eine **gewichtete Klausel**. (Das Gewicht ω bestimmt die Größe der Klausel.)

$(\varGamma, \sigma)$ ist ein **Gegenbeispiel** in $\mathcal{A} \in$ DMod(spec), falls $\mathcal{A} \not\models \sigma(\varGamma)$. Dabei ist σ eine induktive Substitution $\sigma : V^C \to \text{Term}(C)$.

$(\varGamma, \sigma)$ ist ein **Gegenbeispiel in spec**, falls $\mathcal{A} \not\models \sigma(\varGamma)$ für ein $\mathcal{A} \in$ DMod(spec).

Lemma 4.6

Sei $\mathcal{A} \in$ DMod(spec). Es gilt $\mathcal{A} \models \varGamma$ gdw es gibt kein Gegenbeispiel $(\varGamma, \sigma)$ in $\mathcal{A}$.

DMod(spec) $\models \varGamma$ gdw es gibt kein Gegenbeispiel $(\varGamma, \sigma)$ in spec.

Form der Inferenzregeln

Bezeichnung

Ein **Lemma** ist eine Klausel $\varDelta$ mit DMod(spec) $\models \varDelta$. Es ist z. B. jede definierende Gleichung aus E ein Lemma.

Definition 4.16

Eine Inferenzregel hat die Form

$$\frac{\langle \varGamma; \omega \rangle}{\langle \varGamma_1; \omega_1 \rangle, \ldots, \langle \varGamma_n; \omega_n \rangle} \quad \begin{array}{l} \text{mit } \langle \varPi_1; \hat{\omega}_1 \rangle^{U_1}, \ldots, \langle \varPi_k; \hat{\omega}_k \rangle^{U_k} \\ \text{falls Anwendungsbedingung} \end{array}$$

Intention

Die Inferenzregel wird (applikativ) mit den $\langle \varPi_j; \hat{\omega}_j \rangle$ auf das Teilziel $\langle \varGamma; \omega \rangle$ angewandt und erzeugt die neuen Teilziele $\langle \varGamma_i; \omega_i \rangle$. Dabei ist $\varPi_j$ ein Lemma (dann ist $U_j = \mathcal{L}$) oder ein anderes Teilziel, das als Induktionsvoraussetzung angewendet wird (dann ist $U_j = \mathcal{I}$). Ist $k > 0$, so ist die Regel applikativ. Ist $k = 0$, so ist sie nicht applikativ.

Beachte

Im Fall $n = 0$ wird kein neues Teilziel erzeugt. Wird eine Inferenzregel auf $\langle \varGamma; \omega \rangle$ angewandt, so ist $\langle \varGamma; \omega \rangle$ bearbeitet. Es müssen dann alle neuen Teilziele bearbeitet werden.

Eine Ordnung auf Gegenbeispielen (Motivation)

Definition 4.17

Sei $\mathcal{A} \in \mathsf{DMod}(\mathsf{spec})$. Sei Γ eine Klausel und σ eine induktive Substitution. Ein $\mathcal{A}$-**Gegenbeispiel** ist (Γ, σ) mit $\mathcal{A} \not\models \sigma(\Gamma)$, d. h., $\sigma(\Gamma)$ gilt in $\mathcal{A}$ nicht.

Bemerkung

Wir definieren später eine Noethersche Ordnung $\succeq_{\mathcal{A}}$ auf potentiellen $\mathcal{A}$-Gegenbeispielen. Diese wird bezüglich gewichteter Klauseln $\langle \Gamma; \omega \rangle$ definiert, weil das Gewicht ω im wesentlichen die Größe des Gegenbeispiels definiert.

Sei also

$\preceq_{\mathcal{A}}$ eine Ordnung auf den $(\langle \Gamma; \omega \rangle, \sigma)$.

Korrekte Inferenzregeln

Beachte

Die Inferenzregeln sollen Gegenbeispiele verkleinern. Dies spiegelt sich in folgender Definition wider.

Definition 4.18

Eine Inferenzregel

$$\frac{\langle \Gamma; \omega \rangle}{\langle \Gamma_1; \omega_1 \rangle, \ldots, \langle \Gamma_n; \omega_n \rangle} \quad \begin{array}{l} \text{mit } \langle \Pi_1; \hat{\omega}_1 \rangle^{U_1}, \ldots, \langle \Pi_k; \hat{\omega}_k \rangle^{U_k} \\ \text{falls Anwendungsbedingung} \end{array}$$

heißt **korrekt**, falls für jedes $\mathcal{A} \in \mathsf{DMod}(\mathsf{spec})$ und jedes $\mathcal{A}$-Gegenbeispiel der Form (Γ, σ) (1), (2) oder (3) gilt.

(1) Es gibt ein $1 \leq i \leq n$ und ein $\mathcal{A}$-Gegenbeispiel der Form (Γ_i, τ), so dass $(\langle \Gamma_i; \omega_i \rangle, \tau) \preceq_{\mathcal{A}} (\langle \Gamma; \omega \rangle, \sigma)$.

(2) Es gibt ein $1 \leq j \leq k$, so dass $U_j = \mathcal{L}$ und Π_j nicht gültig in spec ist.

(3) Es gibt ein $1 \leq j \leq k$ und ein $\mathcal{A}$-Gegenbeispiel der Form (Π_j, τ), so dass $U_j = \mathcal{I}$ und $(\langle \Pi_j; \hat{\omega}_j \rangle, \tau) \prec_{\mathcal{A}} (\langle \Gamma; \omega \rangle, \sigma)$.

Flexible Induktionsordnungen

Bemerkung

Eine Induktionsordnung muss für das Beispiel geeignet sein. Einzige Voraussetzung ist, dass sie Noethersch ist. Also: Wie findet man geeignete Induktionsordnungen?

Die LPO und die RPO sind zwar starke Ordnungen, sie reichen aber häufig nicht aus, da sie rein syntaktisch definiert sind.

Ziel: Semantische Ordnungen benutzen.

Beispiel Quicksort

$$\begin{aligned} E: \ & \mathsf{qsort}(\mathsf{nil}) = \mathsf{nil} \\ & \mathsf{qsort}(x.l) = \mathsf{app}(\mathsf{qsort}(\mathsf{lower}(x,l)), x.\mathsf{qsort}(\mathsf{upper}(x,l))) \end{aligned}$$

Ziel: $\mathsf{sorted}(\mathsf{qsort}(l)) = \mathsf{true}$.

Ein Induktionsbeweis verlangt eine Induktionsordnung, in der $\mathsf{qsort}(x.l)$ größer als $\mathsf{qsort}(\mathsf{lower}(x,l))$ und $\mathsf{qsort}(\mathsf{upper}(x,l))$ ist.

Idee: Wähle $|l|_{\mathsf{sem}}$ = Länge der Liste l.
Zeige $|x.l|_{\mathsf{sem}} > |\mathsf{lower}(x,l)|_{\mathsf{sem}}$.
$|x.l|_{\mathsf{sem}} > |\mathsf{upper}(x,l)|_{\mathsf{sem}}$.

Semantische Ordnung auf $\mathcal{A}$

Sei $\mathcal{A} \in \mathsf{DMod}(\mathsf{spec})$.

Beachte

Da spec zulässig ist und die Konstruktoren frei sind, gibt es zu jedem $a \in A^C$ genau ein $t_a \in \mathsf{Term}(C)$ mit $a = t_a^{\mathcal{A}}$.

Sei $|t_a|$ die Länge von t_a.

Ist $t \in \mathsf{Term}(F)$ und gilt $\mathcal{A} \models \mathrm{def}(t)$, (d. h., t ist in $\mathcal{A}$ definiert), so gibt es genau ein $t_a \in \mathsf{Term}(C)$ mit $t^{\mathcal{A}} = t_a^{\mathcal{A}}$.

Definition 4.19

Die **semantische $\mathcal{A}$-Länge** von $t \in \mathsf{Term}(F)$ ist

$$|t|_{\mathsf{sem}} = \begin{cases} |t_a| & \text{falls } t^{\mathcal{A}} = t_a^{\mathcal{A}}, t_a \in \mathsf{Term}(C) \\ \text{nicht definiert} & \text{sonst} \end{cases}$$

Die **semantische Ordnung** $\succeq_{\mathcal{A}}$ (bzw. $\succ_{\mathcal{A}}$) auf $\mathsf{Term}(F,V)$ ist gegeben durch

$t_1 \succeq_{\mathcal{A}} t_2$ (bzw. $t_1 \succ_{\mathcal{A}} t_2$) $\ \mathrm{gdw}\ |\sigma(t_1)|_{\mathsf{sem}} \geq |\sigma(t_2)|_{\mathsf{sem}}$ (bzw. $|\sigma(t_1)|_{\mathsf{sem}} > |\sigma(t_2)|_{\mathsf{sem}}$) und $\mathcal{A} \models \mathrm{def}(\sigma(t_i))$ für $i = 1, 2$ und für alle induktiven Substitutionen σ.

Lemma 4.7

$\succ_{\mathcal{A}}$ ist eine Noethersche Partialordnung auf $\mathsf{Term}(F,V)$.

Gilt $t_1 \succeq_{\mathcal{A}} t_2$, so gilt auch $\sigma(t_1) \succeq_{\mathcal{A}} \sigma(t_2)$ für jede Konstruktorsubsitution σ. Analoges gilt für $\succ_{\mathcal{A}}$ statt $\succeq_{\mathcal{A}}$.

Beispiel zur semantischen $\mathcal{A}$-Länge

Sei $F = \{0, s, \mathsf{nil}, ., \mathsf{app}\}$, $C = \{0, s, \mathsf{nil}, .\}$

E : $\mathsf{app}(\mathsf{nil}, l_2) = l_2$
$\mathsf{app}(x.l_1, l_2) = x.\mathsf{app}(l_1, l_2)$

Für $t \in \mathsf{Term}(C)$ gilt $|t|_{\mathsf{sem}} = |t|$,
z. B. $|s^i(0)|_{\mathsf{sem}} = i + 1$
$|s^i(0).s^j(0).\mathsf{nil}|_{\mathsf{sem}} = (i + 1) + (j + 1) + 3$

Für $l_1, l_2 \in \mathsf{Term}(C)$ gilt

$$|\mathsf{app}(l_1, l_2)|_{\mathsf{sem}} = |l_1|_{\mathsf{sem}} + |l_2|_{\mathsf{sem}} - 1.$$

Für jede induktive Subsitution σ und alle $t_1, t_2 \in \mathsf{Term}(F, V)$ gilt

$$|\sigma(\mathsf{app}(t_1, t_2))|_{\mathsf{sem}} = |\sigma(t_1)|_{\mathsf{sem}} + |\sigma(t_2)|_{\mathsf{sem}} - 1.$$

Für alle σ gilt $|\sigma(x.l)|_{\mathsf{sem}} > |\sigma(l)|_{\mathsf{sem}}$ also
$x.l \succ_{\mathcal{A}} l$.

Analog gilt

$\mathsf{app}(l_1, l_2) \succeq_{\mathcal{A}} l_i$ für $i = 1, 2$.

Beachte

Ist t ein Konstruktorterm und $t >_{TT} s$, so gilt $t \succ_{\mathcal{A}} s$.

Eine Ordnung auf den $(\langle \Gamma; \omega \rangle, \sigma)$

Sei $\mathcal{A} \in \mathsf{DMod}(\mathsf{spec})$ wie stets.

Definition 4.20

Sei $k_0 \in \mathbb{N}$ fest vorgegeben.

Ein **Gewicht** $\omega = (t_1, \ldots, t_k)$ ist ein k-Tupel $\omega \in \mathsf{Term}(F, V)^k$ mit $k \leq k_0$ und $\mathsf{DMod}(\mathsf{spec}) \models \mathrm{def}(t_i)$.

Sei $\succ^*_{\mathcal{A}}$ eine Erweiterung von $\succ_{\mathcal{A}}$ auf die Gewichte $\omega \in \mathsf{Term}(F, V)^k$, etwa die Multimengen-Erweiterung $\succ\!\!\succ_{\mathcal{A}}$ oder die lexikographische Erweiterung $\succ^{\mathsf{lex}}_{\mathcal{A}}$.

Eine Ordnung auf den Gewichten ist von der Form $\succ_{G,\mathcal{A}} = \succ^*_{\mathcal{A}}$.

Eine **Ordnung auf den** $(\langle \Gamma; \omega \rangle, \sigma)$ ist von der Form

$$(\langle \Gamma_1; \omega_1 \rangle, \sigma_1) \succeq_{\mathcal{A}} (\langle \Gamma_2; \omega_2 \rangle, \sigma_2) \text{ gdw } \sigma_1(\omega_1) \succeq_{G,\mathcal{A}} \sigma_2(\omega_2)$$

Lemma 4.8

$\succeq_{\mathcal{A}}$ ist eine Noethersche Partialordnung auf den $(\langle \Gamma; \omega \rangle, \sigma)$.

Beachte

Damit ist $\succeq_{\mathcal{A}}$ auch eine Noethersche Partialordnung auf den $\mathcal{A}$-Gegenbeispielen $(\langle \Gamma; \omega \rangle, \sigma)$.

Beispiel für einen Induktionsbeweis

Sei $F = \{0, s, \text{true}, \text{false}, \leq\}$, $C = \{0, s, \text{true}, \text{false}\}$.

$E : 0 \leq y = \text{true} \qquad s(x) \leq 0 = \text{false}$
$\quad s(x) \leq s(y) = x \leq y$

$\Gamma : x \leq x = \text{true}$

Struktur eines Beweises

Das 1. Teilziel ist

(1) $\langle x \leq x = \text{true}; x\rangle$

Das Induktionsschema liefert aus (1)

(2) $\langle 0 \leq 0 = \text{true}; 0\rangle$

(3) $\langle s(x) \leq s(x) = \text{true}; s(x)\rangle$

Abarbeitung von (2) mit der Regel Subsumption und $\langle \Pi; \omega\rangle^{\mathcal{L}} = \langle 0 \leq y = \text{true}; y\rangle^{\mathcal{L}}$ liefert kein neues Teilziel.

Abarbeitung von (3) mit der Regel Rewriting und $\langle \Pi; \omega\rangle^{\mathcal{L}} = \langle s(x) \leq s(y) = x \leq y; x\rangle^{\mathcal{L}}$ liefert (4).

(4) $\langle x \leq x = \text{true}; s(x)\rangle$

Abarbeitung von (4) mit der Regel Subsumption und $\langle \Pi; \omega\rangle^{\mathcal{I}} = (1)^{\mathcal{I}}$ liefert kein neues Teilziel. Beachte hierzu $s(x) \succ_{\mathcal{A}} x$, also darf (1) als Induktionsvoraussetzung angewendet werden.

Damit ist der Beweis erfolgreich abgeschlossen: Alle Teilziele sind abgearbeitet.

Ordnungsliterale

Bemerkung

Im letzten Beispiel war die Voraussetzung $s(x) \succ_{\mathcal{A}} x$ für die Anwendbarkeit der Klausel $\langle x \leq x = \text{true}; x\rangle$ auf $\langle x \leq x = \text{true}; s(x)\rangle$ offensichtlich erfüllt. Dies ist i. Allg. nicht der Fall. Im Beispiel qsort benötigt man die Voraussetzungen $x.l \succ_{\mathcal{A}} \text{upper}(x, l)$ und $x.l \succ_{\mathcal{A}} \text{lower}(x, l)$. Dies wird als Beweisaufgabe aufgefasst.

Also: Man benötigt Ordnungsliterale.

Definition 4.21

Ein **Ordnungsatom** hat die Form $\omega_1 < \omega_2$, wobei die ω_i Gewichte sind.

$\mathcal{A} \in \text{DMod(spec)}$ erfüllt $\omega_1 < \omega_2$, falls für jede induktive Subsitution σ gilt $\sigma(\omega_1) \prec_{G,\mathcal{A}} \sigma(\omega_2)$.

Also

Es gibt Gleichheits-, def- und Ordnungsliterale. Das Symbol $<$ wird in $\mathcal{A}$ als kleiner bezüglich der $\mathcal{A}$-Länge interpretiert.

Bemerkung zur Induktionsordnung (1)

Bemerkung

Wir verwenden als Induktionsordnung die Ordnung $\succeq_{\mathcal{A}}$ auf den $(\langle \Gamma; \omega \rangle, \sigma)$. Sie hängt laut Definition von $\mathcal{A}$ ab. Es soll aber die Gültigkeit einer Klausel in allen $\mathcal{A} \in$ DMod(spec) gezeigt werden.

Also: Wie kann man über alle $\mathcal{A} \in$ DMod(spec) gleichzeitig schließen?

Beachte

spec ist eine zulässige Spezifikation über freien Konstruktoren. Daraus folgt:

a) Gilt $\mathcal{A}, \mathcal{B} \in$ DMod(spec), so sind die C-Redukte $\mathcal{A}^C$ und $\mathcal{B}^C$ vermöge $\varphi(t^{\mathcal{A}}) = t^{\mathcal{B}}$, $t \in$ Term(C), isomorph. Also gilt $|t|^{\mathcal{A}}_{\text{sem}} = |t|^{\mathcal{B}}_{\text{sem}}$ für alle $t \in$ Term(C).

b) Sei $t \in$ Term(F, V). Gilt DMod(spec) $\models \mathrm{def}(t)$, so gibt es zu jeder induktiven Substitution σ genau ein $t_0 \in$ Term(C) mit DMod(spec) $\models \sigma(t) = t_0$. Also gilt auch hier $|t|^{\mathcal{A}}_{\text{sem}} = |t|^{\mathcal{B}}_{\text{sem}}$.

c) Gilt DMod(spec) $\models w_1 < w_2$, so gilt DMod(spec) $\models \mathrm{def}(t)$ für alle t, die in w_1 bzw. w_2 vorkommen.

Bemerkung zur Induktionsordnung (2)

Lemma 4.9

Sei $\omega_1 < \omega_2$ ein Ordnungsatom. Gilt DMod(spec) $\models \omega_1 < \omega_2$, so gilt $(\langle \Gamma_1; \omega_1 \rangle, \sigma) \prec_{\mathcal{A}} (\langle \Gamma_2; \omega_2 \rangle, \sigma)$ für alle $\mathcal{A} \in$ DMod(spec).

Beweis: Dies folgt aus dem letzten Beachte.

Also

Da die Gewichte nur aus definierten Termen aufgebaut sind, ist die Induktionsordnung $\succeq_{\mathcal{A}}$ auf den $(\langle \Gamma; \omega \rangle, \sigma)$ für alle $\mathcal{A}$ gleich. Dies erlaubt die Beweisführung für alle $\mathcal{A} \in$ DMod(spec) gleichzeitig.

Beachte

Der Nachweis der Gültigkeit von $\omega_1 < \omega_2$ in DMod(spec) kann eine komplizierte Beweisaufgabe sein.

Aber: Gilt $\omega_1 \equiv t_1, \omega_2 \equiv t_2$ und $t_1 <_{TT} t_2$ und $t_1, t_2 \in$ Term(C), so ist $\omega_1 < \omega_2$ trivialerweise gültig.

Sichere Inferenzregeln: Motivation

Bemerkung

Anschaulich heißt eine Inferenzregel sicher, wenn sie keine neue Gegenbeispiele einführt.

Es ist manchmal durchaus sinnvoll, unsichere Regeln zu betrachten, z. B. die Generalisierung:

Spezifiziert seien die natürlichen Zahlen mit $+$ und $\cdot$ als definierte Operatoren. Zu zeigen sei $x \cdot x + y \cdot y = y \cdot y + x \cdot x$. Dies kann man generalisieren zur Behauptung $x + y = y + x$. Aus ihr folgt die eigentliche Behauptung, sie ist aber viel leichter zu zeigen.

Für die Lemma-Spekulation und -Verarbeitung gibt es in QUODLIBET einen eigenen Mechanismus.

Sichere Inferenzregeln: Definition

Definition 4.22

$$\frac{\langle \Gamma; \omega \rangle}{\langle \Gamma_1; \omega_1 \rangle, \ldots, \langle \Gamma_n; \omega_n \rangle} \quad \text{mit } \langle \Pi_1; \hat{\omega}_1 \rangle^{U_1}, \ldots, \langle \Pi_k; \hat{\omega}_k \rangle^{U_k} \text{ falls Anwendungsbedingung}$$

heißt **sicher**, falls gilt: Sind Γ und die Π_i gültig in DMod(spec), so ist auch jedes Γ_j gültig in DMod(spec).

Beachte

Ist Γ gültig und $\Gamma \subseteq \Gamma_j$, so ist auch Γ_j gültig.

Lemma 4.10

Gilt $\Gamma \subseteq \Gamma_j$ für alle $j = 1, \ldots, n$, so ist die Inferenzregel sicher.

Bemerkung

Fast alle Inferenzregeln von QUODLIBET sind sicher aufgrund dieses Lemmas.

Ist $\Gamma_j = \Gamma \cup \Lambda_j$, so reicht es zu zeigen, dass ein $\lambda \in \Gamma_j$ gültig ist. Häufig ist es sinnvoll, dazu ein $\lambda \in \Lambda_j$ zu wählen. Die $\lambda' \in \Gamma$ dienen nur zur „Sicherung“, falls kein $\lambda \in \Lambda_j$ gültig ist.

Verallgemeinerte Substitutionen

Erinnerung

Für eine Substitution σ gilt $\sigma(x) \in \mathsf{Term}(C, V^C)$ für alle $x \in V^C$.

Es ist sinnvoll, auch Substitutionen mit $\sigma(x) \in \mathsf{Term}(F, V^C)$ zuzulassen, aber nur dann, wenn $\sigma(x)$ definiert ist, d. h., wenn DMod(spec) $\models \mathrm{def}(\sigma(x))$ gilt.

Definition 4.23

$\sigma : V^C \rightarrow \mathsf{Term}(F, V^C)$ heißt **verallgemeinerte Substitution**, falls DMod(spec) $\models \mathrm{def}(\sigma(x))$ für alle $x \in V^C$ gilt.

Also

Benutzt man in einer Regelanwendung eine verallgemeinerte Subsubstitution, so muss man jedes x mit $\sigma(x) \notin \mathsf{Term}(C, V^C)$ durch das Literal $\mathrm{def}(\sigma(x))$ bewachen. Dieses Literal lässt sich leicht beweisen, falls $\sigma(x) \in \mathsf{Term}(C \cup F^d, V^C)$ ist, wobei F^d aus allen total definierten Funktionssymbolen $f \in F$ besteht. QUODLIBET testet für die Spezifikation der $f \in F$, ob f total definiert ist.

4.4 Die Inferenzregeln

Beachte

Ein Ziel hat die Form $\langle \Gamma; \omega \rangle$ mit

$$\Gamma = (\lambda_1, \ldots, \lambda_n) \qquad \omega = (t_1, \ldots, t_m)$$

Also: Γ ist eine Folge von Literalen.

Regeln transformieren Ziele in Teilziele.

Wir benötigen Regeln für

a) das (triviale) Säubern von Klauseln,
b) die Behandlung von $=$ und $\neq$,
c) die Behandlung von def,
d) die Behandlung von Gewichten, also von $<$,
e) das Anwenden von sicherem Wissen, also eines Axioms oder Lemmas,
f) das Anwenden von Induktionsvoraussetzungen,
g) die Durchführung von Fallunterscheidungen.

Bezeichnung

$=_{\mathsf{lit}}$ ist die Gleichheit auf Literalen.

Beispiele

Sei $\lambda_1 \equiv A$, $\lambda_2 \equiv \neg A$, A ein Atom. Es ist $\lambda_1 =_{\mathsf{lit}} \overline{\lambda_2}$.

Sei $\lambda_1 \equiv s = t$, $\lambda_2 \equiv t = s$. Es ist $\lambda_1 =_{\mathsf{lit}} \lambda_2$.

Einfache Inferenzregeln: Tautologien

Beachte Tautologische Klauseln sind gültig.

Regeln

a) Komplementäre Literale

$$\frac{\langle \Gamma, \lambda, \Delta, \lambda', \Pi; \omega\rangle}{} \quad \text{falls } \lambda' =_{\mathsf{lit}} \overline{\lambda}$$

b) Allgemeingültige Ungleichheit

$$\frac{\langle \Gamma, t_1 \neq t_2, \Delta; \omega\rangle}{} \quad \text{falls } t_1, t_2 \in \mathsf{Term}(C, V^C) \text{ nicht unifizierbar}$$

c) Allgemeingültige Ordnungsatome

$$\frac{\langle \Gamma, (\,) < (u_1, \ldots, u_k), \Delta; \omega\rangle}{} \quad \text{falls } k > 0$$

Beispiele

Gegeben sei das Ziel $\langle \Gamma; \omega\rangle$.
Γ ist gültig, falls Γ
(a) $t_1 = t_2$ und $t_2 \neq t_1$ oder
(b) $f(a, t_1) \neq f(b, t_2)$ mit $f, a, b \in C$, $a \not\equiv b$ und
$t_1, t_2 \in \mathsf{Term}(C, V^C)$ oder
(c) $(\,) < s(0)$
enthält. Dann erzeugt die Abarbeitung von $\langle \Gamma; \omega\rangle$ kein neues Ziel.

Einfache Inferenzregeln: Zerlegung von Atomen

Ziel

Zerlegen von A in $A_1, \ldots, A_k$, so dass A gilt gdw $A_1 \wedge \cdots \wedge A_k$ gilt.

Regeln

a) Zerlegung von Gleichheitsatomen

$$\frac{\langle \Gamma, t_1 = t_2, \Delta; \omega\rangle}{\langle u_1 = v_1, \Gamma, t_1 = t_2, \Delta; \omega\rangle \cdots \langle u_k = v_k, \Gamma, t_1 = t_2, \Delta; \omega\rangle}$$

falls $t_1 \equiv f(u_1, \ldots, u_k), t_2 \equiv f(v_1, \ldots, v_k)$

b) Zerlegung von def-Atomen: Sei $t \equiv f(t_1, \ldots, t_k)$

$$\frac{\langle \Gamma, \mathrm{def}(t), \Delta; \omega\rangle}{\langle \mathrm{def}(t_1), \Gamma, \mathrm{def}(t), \Delta; \omega\rangle \cdots \langle \mathrm{def}(t_k), \Gamma, \mathrm{def}(t), \Delta; \omega\rangle}$$

falls $f \in C \cup V^C$

c) Zerlegung von $<$-Atomen: Sei $\lambda \equiv t < f_1(\cdots f_k(t)), k > 0$

$$\frac{\langle \Gamma, t < f_1(\cdots f_k(t) \cdots), \Delta; \omega\rangle}{\langle \mathrm{def}(t), \Gamma, \lambda, \Delta; \omega\rangle} \quad \text{falls alle } f_i \in C$$

Beispiel: Zerlegen von def-Atomen

Sei s, cons $\in C$ und $t \in F$ und $x, y, z, l \in V^C$

a) Betrachte $t \equiv s^i(x)$
   $\langle \mathrm{def}(x); \omega \rangle$ liefert kein neues Teilziel, ist also gültig.
   $\langle \mathrm{def}(s(x)); \omega \rangle$ liefert $\langle \mathrm{def}(x); \omega \rangle$ als Teilziel, ist also gültig.
   Induktion nach i zeigt $\langle \mathrm{def}(s^i(x)); \omega \rangle$ ist gültig.

b) Betrachte $t \equiv \mathsf{cons}(x + y, \mathsf{cons}(z, l))$
   $\langle \mathrm{def}(t); \omega \rangle$ liefert die beiden Teilziele
   (1) $\langle \mathrm{def}(x + y), \mathrm{def}(t); \omega \rangle$
   (2) $\langle \mathrm{def}(\mathsf{cons}(z, l)), \mathrm{def}(t); \omega \rangle$
   Das Teilziel (1) muss mit dem Lemma „$\mathrm{def}(x + y)$ ist gültig“ gelöst werden.
   Das Teilziel (2) liefert die Teilziele
   (2.1) $\langle \mathrm{def}(z), \mathrm{def}(\mathsf{cons}(z, l)), \mathrm{def}(t); \omega \rangle$
   (2.2) $\langle \mathrm{def}(l), \mathrm{def}(\mathsf{cons}(z, l)), \mathrm{def}(t); \omega \rangle$
   Sie liefern keine neuen Teilziele, sind also gültig.

Beispiel: Zerlegen von $<$-Atomen

Sei $s \in C, t \in \mathsf{Term}(C, V^C)$

a) Betrachte $\lambda \equiv t < s^i(t), \quad i > 0$
   $\langle t < s^i(t); \omega \rangle$ liefert das Teilziel $\langle \mathrm{def}(t), \lambda; \omega \rangle$
   Also ist $\langle x < s^i(x); \omega \rangle$ gültig für alle $i > 0$.

b) Betrachte $\lambda \equiv \mathsf{cons}(x, l) < \mathsf{cons}(x, \mathsf{cons}(y, l))$
   Offenbar gilt $\langle \Gamma; \omega \rangle$ für $\Gamma \equiv \{\lambda\}$.
   Aber die angebene Inferenzregel ist zu schwach. Sie wird später verschärft.

Beachte

Ist $\langle t_1 < t_2; \omega \rangle$ als gültig nachweisbar, so ist auch $\langle \mathrm{def}(t_i); \omega \rangle$, $i = 1, 2$, gültig. Es gibt dann zu jeder Grundinstanz $(\sigma(t_1), \sigma(t_2))$ zwei Terme $t'_1, t'_2 \in \mathsf{Term}(C)$ mit $\sigma(t_i) =_E t'_i$ und $|t'_1| < |t'_2|$. Die intuitive Begründung für $|t'_1| < |t'_2|$ in der Inferenzregel „Zerlegen von $<$-Atomen“ ist, dass $f_i \in C$ für alle i gefordert wird. Daher folgt aus $t =_E t'$ auch $f_1(\ldots f_k(t) \ldots) =_E f_1(\ldots f_k(t') \ldots)$ und $|t'| < |f_1(\ldots f_k(t') \ldots)|$.

Einfache Inferenzregeln: Redundante Literale

Beachte

Ist λ unerfüllbar oder redundant in Γ, so gilt $\langle \Gamma; \omega \rangle$ gdw $\langle \Gamma, \lambda; \omega \rangle$.

Regeln

a) Redundante Literale

$$\frac{\langle \Gamma, \lambda, \Delta, \lambda', \Pi; \omega \rangle}{\langle \Gamma, \lambda, \Delta, \Pi; \omega \rangle} \quad \text{falls } \lambda =_{\text{lit}} \lambda'$$

b) Unerfüllbares Gleichheitsatom

$$\frac{\langle \Gamma, t_1 = t_2, \Delta; \omega \rangle}{\langle \Gamma, \Delta; \omega \rangle} \quad \begin{array}{l} \text{falls } t_1, t_2 \in \text{Term}(C, V^C) \text{ und} \\ t_1, t_2 \text{ nicht unifizierbar} \end{array}$$

c) Redundantes negiertes Gleichheitsatom

$$\frac{\langle \Gamma, t_1 \neq t_2, \Delta; \omega \rangle}{\langle \Gamma, \Delta; \omega \rangle} \quad \begin{array}{l} \text{falls } t_i \equiv f(u_1^i, \ldots, u_k^i), i = 1, 2 \\ u_j^1 \neq u_j^2 \in \Gamma \cup \Delta \text{ für alle } 1 \leq j \leq k \end{array}$$

d) Unerfüllbares Ordnungsliteral

$$\frac{\langle \Gamma, (u_1, \ldots, u_k) < (\,), \Delta; \omega \rangle}{\langle \Gamma, \Delta; \omega \rangle} \quad \text{falls } k > 0$$

Ausnutzen negativer Literale

Beachte

Eine Klausel $\Gamma = \neg A_1 \vee \cdots \vee \neg A \vee B_1 \vee \cdots \vee B_m$ kann man auch schreiben als

$$A_1 \wedge \cdots \wedge A_n \Rightarrow B_1 \vee \cdots \vee B_m$$

Also: Ist $A_1 \equiv s = t$, so kann man in jedem B_i jedes s durch t ersetzen.

Regeln

a) Constant Rewriting

$$\frac{\langle \Gamma, \lambda[t_1]_p, \Delta; \omega \rangle}{\langle \Gamma, \lambda[t_2]_p, \Delta; \omega \rangle} \quad \text{falls } t_1 \neq t_2 \text{ in } \Gamma \cup \Delta$$

b) $\neq$-Unifikation

$$\frac{\langle \Gamma, t_1 \neq t_2, \Delta; \omega \rangle}{\langle \tau(\Gamma), \tau(\Delta); \tau(\omega) \rangle} \quad \begin{array}{l} \text{falls } \tau = \text{mgu}(t_1, t_2), \\ t_1, t_2 \in \text{Term}(C, V^C) \end{array}$$

Beispiele Sei $s(x) \in \text{Term}(C, V^C)$

a) $$\frac{\langle t \neq 0 \vee f(x, t) = f(x, 0); \omega \rangle}{\langle t \neq 0 \vee f(x, 0) = f(x, 0); \omega \rangle} \quad \text{mit Constant Rewriting}$$

b) $$\frac{\langle s(x) \neq s(y), x = y; \omega \rangle}{\langle x = x; \omega \rangle} \quad \text{mit } \neq\text{-Unifikation}$$

In beiden Fällen kann das neue Teilziel durch (wiederholte) Anwendung der Inferenzregel „Zerlegung von Gleichheitsatomen“ gezeigt werden.

Fallunterscheidung nach einer Klausel (1)

Beachte

Man kann eine Fallunterscheidung nach jeder allgemeingültigen Klausel machen.

Beispiel: $\Delta = \{\lambda, \overline{\lambda}\}$ ist stets allgemeingültig. Die Fallunterscheidung mit Δ angewandt auf Γ liefert
$\Gamma \cup \{\neg\lambda\}$ und $\Gamma \cup \{\lambda\}$ also $\lambda \Rightarrow \Gamma$ und $\neg\lambda \Rightarrow \Gamma$

Diese 2 Teilziele sind häufig einfacher zu beweisen.

Ansatz hier

Fallunterscheidung nach beliebigem Δ

Sei $\Delta = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$
$\Delta_i = \{\overline{\lambda_i}, \lambda_{i-1}, \ldots, \lambda_1\}$ für $i = 1, \ldots, n$

Dann ist $\Delta_0 \equiv \Delta \cup \Delta_1 \cup \cdots \cup \Delta_n$ allgemeingültig.

Anwendung der Fallunterscheidung auf $\langle \Gamma; \omega \rangle$ mit Δ_0 liefert die Teilziele

$$\langle \Gamma \cup \Delta_1; \omega \rangle \langle \Gamma \cup \Delta_2; \omega \rangle \cdots \langle \Gamma \cup \Delta_n; \omega \rangle \langle \Gamma \cup \Delta; \omega \rangle$$

Bezeichnung

$\Delta_1, \ldots, \Delta_n, \Delta$ ist die Fallunterscheidung zu Δ.

Fallunterscheidung nach einer Klausel (2)

Sei $\Delta = (\lambda_1, \lambda_2, \ldots, \lambda_n)$
$\Delta_i = (\overline{\lambda_i}, \lambda_{i-1}, \ldots, \lambda_1)$ für $i = 1, \ldots, n.$

Regel Fallunterscheidung nach Δ

$$\frac{\langle \Gamma; \omega \rangle}{\langle \Delta_1, \Gamma; \omega \rangle \cdots \langle \Delta_n, \Gamma; \omega \rangle \quad \langle \Delta, \Gamma; \omega \rangle}$$

Beispiel

$\Delta = \{\mathsf{less}(x, y) = \mathsf{true}, x = y, \mathsf{less}(y, x) = \mathsf{true}\}$ liefert

$$\frac{\langle \Gamma; \omega \rangle}{\begin{array}{c} \langle \mathsf{less}(x, y) \neq \mathsf{true}, \Gamma; \omega \rangle \\ \langle x \neq y, \mathsf{less}(x, y) = \mathsf{true}, \Gamma; \omega \rangle \\ \langle \mathsf{less}(y, x) \neq \mathsf{true}, x = y, \mathsf{less}(x, y) = \mathsf{true}, \Gamma; \omega \rangle \\ \langle \Delta, \Gamma; \omega \rangle \end{array}}$$

Beachte

Ist Δ ein Lemma, so ist das letzte Teilziel leicht zu beweisen, z. B. durch Subsumption. Dies gilt z. B., wenn Δ schon bewiesen, eine definierende Gleichung oder allgemeingültig ist.

Fallunterscheidung nach Substitutionen (1)

Beachte

Γ ist gültig genau dann, wenn für jede induktive Substitution $\sigma(\Gamma)$ gültig ist.

Also ist es sinnvoll, alle $\sigma(\Gamma)$ durch eine Fallunterscheidung zu „überdecken".

Definition 4.24

Eine Menge $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ von Konstruktorsubstitutionen heißt eine **Überdeckung** für Γ, falls es zu jeder induktiven Substitution σ ein $\sigma_i \in \Sigma$ und ein τ gibt mit $\sigma(x) = \tau(\sigma_i(x))$ für alle $x \in \mathsf{Var}(\Gamma) \cap V^C$.

Beispiel zu Überdeckungen

Beispiel

Sei $\mathsf{var}(\Gamma) = \{x, y\}$ und $C = \{0, s\}$. Überdeckungen zu Γ sind Σ_1 bis Σ_4 mit

Σ_1:

x	y
0	0
0	$s(y)$
$s(x)$	0
$s(x)$	$s(y)$

Σ_2:

x	y
0	y
$s(x)$	0
$s(x)$	$s(y)$

Σ_3:

x	y
0	y
$s(x)$	y

Σ_4:

x	y
0	y
$s(0)$	y
$s^2(x)$	y

Also: Σ_1 (bzw. Σ_4) besteht aus den 4 (bzw. 3) Substitutionen wie in der Tabelle angegeben.

Fallunterscheidung nach Substitutionen (2)

Sei $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ eine Überdeckung zu Γ.

Regel Fallunterscheidung mit Σ

$$\frac{\langle \Gamma; \omega \rangle}{\langle \sigma_1(\Gamma); \sigma_1(\omega) \rangle \cdots \langle \sigma_n(\Gamma); \sigma_n(\omega) \rangle}$$

Beispiel Ackermann-Funktion

$$\begin{aligned} E : a(0,y) &= s(y) \\ a(s(x),0) &= a(x,s(0)) \\ a(s(x),s(y)) &= a(x,a(s(x),y)) \end{aligned}$$

Zu zeigen sei die Gültigkeit von $\Gamma = \{\mathrm{def}(a(x,y))\}$, also dass a total definiert ist.

Die Definition von a liefert, dass Σ_2 (von Folie 142) eine sinnvolle Überdeckung ist. Also ergibt sich

$$\frac{\langle \mathrm{def}(a(x,y)); \omega(x,y) \rangle}{\langle \mathrm{def}(a(0,y)); \omega(0,y) \rangle \quad \langle \mathrm{def}(a(s(x),0)); \omega(s(x),0) \rangle \quad \langle \mathrm{def}(a(s(x),s(y))); \omega(s(x),s(y)) \rangle}$$

Applikative Inferenzregeln (1)

Beachte

Bisher waren alle Inferenzregeln nicht-applikativ.

Sinn einer applikativen Inferenzregel:

$$\frac{\langle \Gamma; \omega \rangle}{\langle \Gamma_1; \omega \rangle \cdots \langle \Gamma_n; \omega \rangle} \quad \text{mit } \langle \Pi; \omega' \rangle^{\mathcal{U}}$$

Sie wendet das „Wissen“ Π instanziert (d. h. in der Form $\mu(\Pi)$) auf Γ an. Dabei ist das Wissen Π entweder sicher (Anwendung eines Lemmas, dann ist $\mathcal{U} = \mathcal{L}$) oder induktiv (Anwendung einer Induktionshypothese, dann ist $\mathcal{U} = \mathcal{I}$).

Probleme

a) Ist $\mathcal{U} = \mathcal{I}$, so muss $\mu(\omega') < \omega$ sicher gestellt sein.

b) $\mu(\Pi)$ ist nur dann eine zulässige Instanz von Π, wenn $\mu(x) \in \mathrm{Term}(C, V^C)$ für alle $x \in \mathrm{Var}(\Pi) \cap V^C$ ist. Was macht man, wenn dies nicht gilt?

c) Manchmal sind noch nicht alle logischen Voraussetzungen für die Anwendung von Π bewiesen. Wie holt man diese Beweise nach?

Applikative Inferenzregeln (2)

Ziel

Behandlung von verallgemeinerten Substitutionen.

Bezeichnung

Gilt $\mu(x) \in \mathsf{Term}(C, V^C)$ für alle $x \in V^C$, so ist μ eine **Konstruktorsubstitution**. Eine Substitution μ ist „im wesentlichen" eine Konstruktorsubstitution, falls $\mathrm{def}\,\mu(x)$ gültig ist für alle $x \in V^C$. Wir bezeichnen eine solche Substitution als **verallgemeinerte Substitution**.

Definition 4.25

$\mathsf{DefCond}(\mu, \Pi) = \{\neg \mathrm{def}\,\mu(x) \mid x \in \mathsf{Var}(\Pi) \cap V^C, \mu(x) \notin \mathsf{Term}(C, V^C)\}$

Beachte

Es gelte $\mathcal{A} \models \Pi$. Dann gilt für jede Konstruktor- und jede verallgemeinerte Substitution $\mathcal{A} \models \mu(\Pi)$; für beliebige Substitutionen μ gilt dies hingegen nicht, da diese Konstruktorvariablen an undefinierte Terme binden können.

Gilt $\mathcal{A} \models \neg\mathsf{DefCond}(\mu, \Pi)$, so ist μ eine verallgemeinerte Substitution bzgl. Π. Also gilt $\mathcal{A} \models \neg\mathsf{DefCond}(\mu, \Pi) \Rightarrow \mu(\Pi)$ bzw. $\mathcal{A} \models \mathsf{DefCond}(\mu, \Pi) \cup \mu(\Pi)$.

Applikative Inferenzregeln (3)

Beachte

Will man $\mu(\Pi)$ auf Γ anwenden, so müssen einige Voraussetzungen erfüllt sein:

Sei etwa Π von der Form $\Pi \equiv \Pi_0 \vee l = r$, also $\Pi \equiv \neg(\Pi_0) \Rightarrow l = r$, und es soll Γ mit $\mu(\Pi)$ umgeschrieben werden, also mit der Regel $\mu(l) \rightarrow \mu(r)$.

Dann müssen $\neg\mu(\Pi_0)$ und $\neg\mathsf{DefCond}(\mu, \Pi)$ bei der Anwendung auf die Klausel Γ gültig sein. Wie garantiert man dies?

Ansatz

Wir berechnen eine Klausel Θ, die gewährleistet, dass die Bedingungen im Kontext Θ gültig sind:

- $\mu(\Pi_0) \subseteq \Gamma \cup \Theta$ (damit $\neg\mu(\Pi_0)$ gültig ist),
- $\mathsf{DefCond}(\mu, \Pi) \subseteq \Gamma \cup \Theta$ (damit $\mu(\Pi)$ zulässig ist).

Wir wenden die Regel auf Γ im Kontext Θ an, d.h. auf $\Gamma \cup \Theta$.

Da wir Γ jedoch im leeren Kontext zeigen wollen, führen wir eine Fallunterscheidung nach Θ durch. Im letzten Fall dieser Fallunterscheidung ist der Kontext Θ erfüllt und wir können die Regel anwenden. In allen anderen Fällen müssen wir die Gültigkeit von Γ nachweisen, falls der Kontext nicht erfüllt ist. Meist dienen diese Fälle dazu im nachhinein zu zeigen, dass der Kontext von vornherein erfüllt war, indem man die Gültigkeit der durch die Fallunterscheidung neu eingeführten Literale nachweist.

Nicht-induktive Subsumption (1)

Regel

$$\frac{\langle \Gamma ; \omega \rangle}{\langle \Delta_1, \Gamma ; \omega \rangle \cdots \langle \Delta_n, \Gamma ; \omega \rangle} \quad \text{mit } \langle \Pi, \omega' \rangle^{\mathcal{L}}$$

falls es μ und Θ gibt mit
- DefCond$(\mu, \Pi) \cup \mu(\Pi) \subseteq \Gamma \cup \Theta$,
- $\Delta_1, \ldots, \Delta_n, \Delta$ ist die Fallunterscheidung zu $\Delta = \Theta$.

Beachte

a) Sei $\mu(\Pi) \subseteq \Gamma$ und μ eine Konstruktorsubstitution. Dann ist DefCond$(\mu, \Pi) = \emptyset$ und man kann $\Theta = \emptyset$ wählen. Es entsteht kein neues Teilziel. Dies entspricht der klassischen Subsumption.

b) Sonst: Man benötigt Θ mit
- $\mu(\Pi) \subseteq \Gamma \cup \Theta$, damit $\mu(\Pi)$ $\Gamma \cup \Theta$ subsumiert,
- DefCond$(\mu, \Pi) \subseteq \Gamma \cup \Theta$ damit $\mu(\Pi)$ eine zulässige Instanz ist.

Das Teilziel zum Fall Δ muss nicht mehr nachgewiesen werden, da die Klausel $\Gamma \cup \Theta$ von $\mu(\Pi)$ subsumiert wird. Die oben angegebenen Teilziele gewährleisten, dass Γ auch dann gilt, falls der Kontext nicht erfüllt ist.

Gilt $\mu(\Pi) \subseteq \Gamma$ nicht, so spricht man von **partieller Subsumption**.

Nicht-induktive Subsumption (2)

Beispiel

a) Sei $x, x', y, y' \in V^C$, $a \in C$, $f, g \in D = F - C$
$\Pi \equiv x' + y' = y' + x'$
$\Gamma \equiv f(x) + g(a, y) = g(a, y) + f(x)$
$\mu = \{x' \leftarrow f(x), y' \leftarrow g(a, y)\}$

Dann ist DefCond$(\mu, \Pi) = \{\neg \mathrm{def} f(x), \neg \mathrm{def} g(a, y)\}$.
Wähle $\Theta =$ DefCond(μ, Π).
Dann gilt DefCond$(\mu, \Pi) \cup \mu(\Pi) \subseteq \Gamma \cup \Theta$, also

$$\frac{\langle \Gamma ; \omega \rangle}{\langle \mathrm{def} f(x), \Gamma ; \omega \rangle \quad \langle \mathrm{def} g(a, y), \neg \mathrm{def} f(x), \Gamma ; \omega \rangle}$$

Das Lemma $\mathrm{def} f(x)$ würde das erste Teilziel subsumieren und das Lemma $\mathrm{def} g(x, y)$ würde das zweite Teilziel subsumieren.

b) Sei $\Gamma' = \Gamma, \neg \mathrm{def} f(x), \neg \mathrm{def} g(a, y)$.
Dann kann man $\Theta = \emptyset$ wählen und $\langle \Gamma' ; \omega \rangle$ erzeugt kein neues Teilziel.

Nicht-induktive Subsumption (3)

Beispiel

Sei $\Gamma = \{ \lambda, \lambda' \}$ und $\Pi = \{ \lambda_1, \lambda_2, \lambda_3 \}$ ein Lemma und μ eine Konstruktorsubstitution mit $\mu(\lambda_1) = \lambda$. Dann sind gültig:

$$\neg\mu(\lambda_2) \wedge \neg\mu(\lambda_3) \Rightarrow \lambda$$
$$\neg\mu(\lambda_2) \wedge \neg\mu(\lambda_3) \Rightarrow \Gamma$$

Für die Gültigkeit von Γ bleibt $\langle\neg\mu(\lambda_2), \Gamma; \omega)\rangle$ und $\langle\neg\mu(\lambda_3), \Gamma; \omega\rangle$ zu zeigen.

Anwendung der Subsumptionsregel liefert

$$\Delta = \Phi = \{ \mu(\lambda_2), \mu(\lambda_3) \},$$

also

$$\frac{\langle\Gamma; \omega\rangle}{\langle\neg\mu(\lambda_2), \Gamma; \omega\rangle\langle\neg\mu(\lambda_3), \mu(\lambda_2), \Gamma; \omega\rangle}$$

Also: Sind die beiden Teilziele gültig, so ist auch $\langle\Gamma; \omega\rangle$ gültig.

Bemerkungen zur Termersetzung

Seien Π, Γ Klauseln und λ ein Literal, B eine Konjunktion von Literalen (eine Bedingung) und $\mathcal{A}$ ein Datenmodell.

1. $\Pi : B \Rightarrow l = r$ und $\Gamma : B \Rightarrow \lambda[l] \vee \Gamma_1$.
   Es sei $\mathcal{A} \models \Pi$. Dann gilt
   $$\mathcal{A} \models \Gamma \mathrm{gdw} \mathcal{A} \models B \Rightarrow \lambda[r] \vee \Gamma_1.$$

2. $\Pi : B_1 \wedge B_2 \Rightarrow l = r$ und $\Gamma : \mu(B_1) \Rightarrow \lambda[\mu(l)] \vee \Gamma_1$.
   Es sei $\mathcal{A} \models \mu(\Gamma)$. Es gilt
   $$\begin{aligned} &\mathcal{A} \models \Gamma \\ &\quad \mathrm{gdw} \quad \mathcal{A} \models \mu(B_2) \Rightarrow \Gamma \text{ und } \mathcal{A} \models \neg\mu(B_2) \Rightarrow \Gamma \\ &\quad \mathrm{gdw} \quad \mathcal{A} \models \mu(B_1) \wedge \mu(B_2) \Rightarrow \lambda[\mu(r)] \vee \Gamma_1 \text{ und} \\ &\qquad\qquad \mathcal{A} \models \neg\mu(B_2) \Rightarrow \Gamma. \end{aligned}$$

   Schreibt man dies in Klauselform, so ergeben sich die Inferenzregeln für die Termersetzung.

Nicht-induktive Termersetzung (1)

Beachte

Wir wollen Γ mit Π umschreiben: Es gebe ein λ in Γ und ein Atom $l = r$ in Π mit $\lambda/p \equiv \mu(l)$, es soll $\lambda[\mu(l)]_p$ durch $\lambda[\mu(r)]_p$ ersetzt werden.

Logisch hat Π die Form $\neg\Pi_0 \Rightarrow l = r$. Um die Termersetzung durchführen zu können, müssen $\neg\Pi_0$ und $\neg\mathrm{DefCond}(\mu, \Pi)$ bei der Anwendung gültig sein.

Dies geschieht wie eben: Ist μ eine Konstruktorsubstitution und gilt $\mu(\Pi_0) \subseteq \Gamma$, so sind die Voraussetzungen erfüllt. Sonst benötigen wir ein Θ, das den benötigten Kontext erzeugt.

Regel Sei $\Gamma \equiv \Gamma_1, \lambda, \Gamma_2$ und $\Pi \equiv \Pi_1, l \doteq r, \Pi_2$

$$\frac{\langle \Gamma; \omega \rangle}{\langle \Delta_1, \Gamma; \omega \rangle \cdots \langle \Delta_n, \Gamma; \omega \rangle \ \langle \Delta, \Gamma_1, \lambda[\mu(r)]_p, \Gamma_2; \omega \rangle} \quad \text{mit } \langle \Pi; \omega' \rangle^{\mathcal{L}}$$

falls $\lambda/p \equiv \mu(l)$ und

- $\mathrm{DefCond}(\mu, \Pi) \cup \mu(\Pi_1) \cup \mu(\Pi_2) \subseteq \Gamma_1 \cup \Gamma_2 \cup \{\mu(l) = \mu(r)\} \cup \Theta$
- $\Delta_1, \ldots, \Delta_n, \Delta$ ist die Fallunterscheidung zu $\Delta = \Theta$.

Nicht-induktive Termersetzung (2)

Beispiel

a) Sei $\Pi : \neg a = b \vee c = d \vee l = r_1$ (oder äquivalent dazu $a = b \wedge c \neq d \Rightarrow l = r_1$)
$\Gamma : \neg a = b \vee l = r_2$ (oder $a = b \Rightarrow l = r_2$)
Es soll gelten $\{\neg a = b, c = d\} \subseteq \{\neg a = b, l = r_1\} \cup \Theta$.
Setze $\Theta = \{c = d\} = \Delta$, also $n = 1$ und $\Delta_1 = \{\neg c = d\}$
Es gilt

$$\frac{\langle \Gamma; \omega \rangle}{\langle \neg c = d, \Gamma; \omega \rangle \quad \langle c = d, \neg a = b, r_1 = r_2; \omega \rangle}$$

Also: Der Beweis zu $a = b \Rightarrow l = r_2$ reduziert sich auf die Beweise zu
$c = d \wedge a = b \Rightarrow l = r_2$ bzw. $\neg c = d \wedge a = b \Rightarrow r_1 = r_2$

b) Sei $\Pi' \equiv \Gamma$ und $\Gamma' \equiv \Pi$.
Hier subsumiert die Voraussetzung von Γ' die von Π': Es gilt $\{\neg a = b\} \subseteq \{\neg a = b, c = d, l = r_2\}$.
Also kann man $\Theta = \emptyset$ wählen.
Das Teilziel $\langle \Gamma'; \omega \rangle$ produziert nur ein Teilziel, nämlich $\langle \neg a = b, c = d, r_2 = r_1; \omega \rangle$.

Induktive Inferenzregeln

Beachte

Es soll die Induktionsvoraussetzung $\langle \Pi ; \omega' \rangle$ mit Substitution μ auf das Teilziel $\langle \Gamma ; \omega \rangle$ angewendet werden. Es gibt 2 Möglichkeiten: Subsumption und Termersetzung mit Π und μ.

Dazu müssen die bisherigen Bedingungen der Subsumption bzw. Termersetzung gelten und es muss $\mu(\omega') < \omega$ gelten.

Beispiel

Zu beweisen sei $\langle x + y = y + x ; y \rangle \equiv \langle \Pi ; \omega' \rangle$.

Dies führe zum Teilziel $\langle x + y = y + x ; s(y) \rangle \equiv \langle \Pi ; \omega \rangle$.

Dies kann man unterschiedlich behandeln:

i) Es subsumiert $\langle \Pi ; \omega' \rangle$ induktiv $\langle \Pi ; \omega \rangle$ mit $\mu = \text{id}$.

ii) Die induktive Termersetzung mit $\langle \Pi ; \omega' \rangle$ und $\mu = \text{id}$ überführt $\langle \Pi ; \omega \rangle$ in das triviale Teilziel $\langle x + y = x + y ; \omega \rangle$.

Beachte: In beiden Fällen gilt $\mu(\omega') < \omega$.

Induktive Subsumption

Beachte

Die Beachtung der Bedingung $\mu(\omega') < \omega$ überführt die Regel zur nicht-induktiven Subsumption in die folgende Regel.

Regel

$$\frac{\langle \Gamma ; \omega \rangle}{\begin{array}{c}\langle \Delta_1, \Gamma ; \omega \rangle \cdots \langle \Delta_n, \Gamma ; \omega \rangle \\ \langle \mu(\omega') < \omega, \Delta, \Gamma ; \omega \rangle\end{array}} \quad \text{mit } \langle \Pi ; \omega' \rangle^{\mathcal{I}}$$

falls es μ und Θ gibt mit
- DefCond$(\mu, \Pi) \cup \mu(\Pi) \subseteq \Gamma \cup \Theta$,
- $\Delta_1, \ldots, \Delta_n, \Delta$ ist die Fallunterscheidung zu $\Delta = \Theta$.

Beachte

Das Literal $\mu(\omega') < \omega$ benötigt man nicht in den Teilzielen $(\Delta_i, \Gamma ; \omega\rangle$ für die Zulässigkeit von μ. Sie tritt nur im letzten Teilziel auf.

Beispiel zur induktive Subsumption

Beispiel

Zu beweisen sei
$\langle \Pi; \omega\rangle \equiv \langle x = y, \mathsf{less}(x, y) = \mathsf{true}, \mathsf{less}(y, x) = \mathsf{true}; (x, y)\rangle$

Dies führe zum Teilziel

$$\begin{aligned}\langle \Gamma'; \omega'\rangle \quad &\equiv \quad \langle s(x) = s(y), \mathsf{less}(s(x), s(y)) = \mathsf{true},\\ &\qquad \mathsf{less}(s(y), s(x)) = \mathsf{true}; (s(x), s(y))\rangle\end{aligned}$$

Simplifikation von $\langle \Gamma'; \omega'\rangle$ führt zu
$\langle \Gamma; \omega'\rangle \equiv \langle s(x) = s(y), \mathsf{less}(x, y) = \mathsf{true}, \mathsf{less}(y, x) = \mathsf{true}; \omega'\rangle$

Induktive Termersetzung mit $\langle \Pi; \omega\rangle$, $\Delta = \{\ x = y\ \}$ und $\mu = \mathsf{id}$ führt zu

$$\begin{aligned}\langle \Gamma_1; \omega'\rangle \quad &\equiv \quad \langle \neg x = y, s(x) = s(y), \mathsf{less}(x, y) = \mathsf{true},\\ &\qquad \mathsf{less}(y, x) = \mathsf{true}; \omega'\rangle\\ \langle \Gamma_2; \omega'\rangle \quad &\equiv \quad \langle \omega < \omega', x = y, \Gamma; \omega'\rangle\end{aligned}$$

Das Ziel $\langle \Gamma_1; \omega'\rangle \equiv \langle \neg x = y, s(x) = s(y), \ldots; \omega'\rangle$ führt zu $\langle \Gamma_1'; \omega'\rangle \equiv \langle \neg x = y, s(x) = s(x), \ldots; \omega'\rangle$ und wird wegen des Atoms $s(x) = s(x)$ durch Säubern gelöst. Das Ziel $\langle \Gamma_2; \omega'\rangle$ wird wegen des Atoms $(x, y) < (s(x), s(y))$ auch durch Säubern gelöst.

Damit ist das Teilziel $\langle \Gamma; \omega\rangle$ bewiesen.

Induktive Termersetzung

Beachte

Dies dient zum Umschreiben eines aktuellen Teilziels $\langle \Gamma; \omega\rangle$ mit einer Induktionsvoraussetzung $\langle \Pi; \omega'\rangle$.

Die Übertragung der Regel zur nicht-induktiven Termersetzung läuft wie eben.

Regel Sei $\Gamma \equiv \Gamma_1, \lambda, \Gamma_2$ und $\Pi \equiv \Pi_1, l \doteq r, \Pi_2$

$$\frac{\langle \Gamma; \omega\rangle}{\begin{array}{c}\langle \Delta_1, \Gamma; \omega\rangle \cdots \langle \Delta_n, \Gamma; \omega\rangle\ \langle \Delta, \Gamma_1, \lambda[\mu(r)]_p, \Gamma_2; \omega\rangle\\ \langle \mu(\omega') < \omega, \Delta', \Gamma; \omega\rangle\end{array}}$$

mit $\langle \Pi; \omega'\rangle^{\mathcal{I}}$, falls

- $\lambda/p \equiv \mu(l)$,
- $\mathsf{DefCond}(\mu, \Pi) \cup \mu(\Pi_1) \cup \mu(\Pi_2) \subseteq \Gamma_1 \cup \Gamma_2 \cup \{\mu(l) = \mu(r)\} \cup \Theta$,
- $\Delta_1, \ldots, \Delta_n, \Delta$ ist die Fallunterscheidung zu $\Delta = \Theta$,
- $\Delta' \equiv \mu(l) = \mu(r), \Delta$.

Die Inferenzregeln von QUODLIBET

Bemerkung

Die hier angegebenen Inferenzregeln dienten der Motivation und dem Verständnis, welche Arten von Inferenzregeln benötigt werden. Sie müssen für den praktischen Gebrauch noch verfeinert werden.

Ziel

Angabe aller Inferenzregeln von QUODLIBET.

Dazu

Wir benötigen einige Bezeichnungen.

Bezeichnungen

Minimale Unterscheidungs-Positionen von t_1 und t_2

$\mathsf{MinDifPos}(t_1, t_2)$ besteht aus den minimalen Positionen aus $\{\pi \in \mathsf{Pos}(t_1) \cap \mathsf{Pos}(t_2) \mid \mathsf{top}(t_1/\pi) \neq \mathsf{top}(t_2/\pi)\}$.

Minimale Nicht-Konstruktor-Positionen in t

$\mathsf{MinNonCPos}(t)$ besteht aus den minimalen Positionen aus $\{\pi \in \mathsf{Pos}(t) \mid \mathsf{top}(t/\pi) \notin C \cup V^C\}$.

Die **C-Front** von t

Sei $t \in \mathsf{Term}(F, V)$. Seien $t_1, \ldots, t_m$ die verschiedenen maximalen Teilterme von t mit $\mathsf{top}(t_i) \notin C \cup V^C$. Seien $x_1, \ldots, x_m \in V^C$ neue Variable. Die C-Front $\hat{t}$ von t entsteht durch Ersetzen der t_i durch x_i.

Termmengen

$\mathcal{T}(\mathsf{sig}^C, V^C) = \mathsf{Term}(C, V^C)$.

Literale

$(\lambda_1, \ldots, \lambda_n)[m] = \lambda_m$ für $1 \leq m \leq n$.

Non-Applicative Inference Rules

Establishing Simple Tautologies

Complementary Literals:
`compl-lit` *m n*

$$\frac{\langle \Gamma, \lambda, \Delta, \lambda', \Pi \,;\, w\rangle}{}\qquad \text{if} \quad \begin{array}{l} - \ (\Gamma, \lambda, \Delta, \lambda', \Pi)[m] = \lambda \\ - \ (\Gamma, \lambda, \Delta, \lambda', \Pi)[n] = \lambda' \\ - \ \lambda' =_{\text{lit}} \overline{\lambda}. \end{array}$$

$\neq$-Tautology:
`=/=-taut` *m*

$$\frac{\langle \Gamma,\ t_1 \neq t_2 \,,\Delta \,;\, w\rangle}{}\qquad \text{if} \quad \begin{array}{l} - \ (\Gamma,\ t_1 \neq t_2 \,, \Delta)[m] = \ t_1 \neq t_2 \\ - \ t_1, t_2 \in \mathcal{T}(\mathsf{sig}^C, V^{\mathrm{C}}) \\ - \ t_1 \text{ and } t_2 \text{ are not unifiable.} \end{array}$$

$<$-Tautology:
`<-taut` *m*

$$\frac{\langle \Gamma,\ () < (u_1, \ldots, u_k)\,, \Delta \,;\, w\rangle}{}\qquad \text{if} \quad \begin{array}{l} - \ (\Gamma,\ () < (u_1, \ldots, u_k)\,, \Delta)[m] = \\ \quad\ () < (u_1, \ldots, u_k) \\ - \ k > 0. \end{array}$$

Decomposing Atoms

$=$-Decomposition:
`=-decomp` *m*

$$\frac{\langle \Gamma,\ t_1 = t_2 \,, \Delta \,;\, w\rangle}{\langle\, u_1 = v_1 \,, \Gamma,\ t_1 = t_2 \,, \Delta \,;\, w\rangle \ \ldots \ \langle\, u_k = v_k \,, \Gamma,\ t_1 = t_2 \,, \Delta \,;\, w\rangle}$$

if
- $(\Gamma,\ t_1 = t_2 \,, \Delta)[m] = \ t_1 = t_2$
- $\mathrm{top}(t_1) = \mathrm{top}(t_2)$
- $u_1 = v_1 \,, \ldots,\ u_k = v_k$ are exactly those equations in

  $$\{\, t_1/p \doteq t_2/p \mid p \in \mathrm{MinDifPos}(t_1, t_2) \,\}$$

  whose complements do not occur in Γ, Δ.

def-Decomposition:
`def-decomp` *m*

$$\frac{\langle \Gamma,\ \mathrm{def}(t)\,, \Delta \,;\, w\rangle}{\langle\, \mathrm{def}(u_1)\,, \Gamma,\ \mathrm{def}(t)\,, \Delta \,;\, w\rangle \ \ldots \ \langle\, \mathrm{def}(u_k)\,, \Gamma,\ \mathrm{def}(t)\,, \Delta \,;\, w\rangle}$$

if
- $(\Gamma,\ \mathrm{def}(t)\,, \Delta)[m] = \ \mathrm{def}(t)$
- $\mathrm{top}(t) \in (C \cup V^{\mathrm{C}})$
- $\mathrm{def}(u_1)\,, \ldots,\ \mathrm{def}(u_k)$ are exactly those definedness atoms in

  $$\{\, \mathrm{def}(t/p) \mid p \in \mathrm{MinNonCPos}(t) \,\}$$

  whose complements do not occur in Γ, Δ.

$<$-Decomposition:
`<-decomp` *m*

$$\frac{\langle \Gamma,\ t_1 < t_2 \,, \Delta \,;\, w\rangle}{\langle\, \mathrm{def}(u_1)\,, \Gamma,\ t_1 < t_2 \,, \Delta \,;\, w\rangle \ \ldots \ \langle\, \mathrm{def}(u_k)\,, \Gamma,\ t_1 < t_2 \,, \Delta \,;\, w\rangle}$$

if
- $(\Gamma,\ t_1 < t_2 \,, \Delta)[m] = \ t_1 < t_2$
- $\mathrm{top}(t_2) \in C$
- there are $\hat{t}_1, \hat{t}_2 \in \mathcal{T}(\mathsf{sig}^C, V^{\mathrm{C}})$ such that
  - for $i \in \{1, 2\}$, $\hat{t}_i$ is a C-front for t_i
  - for each $p_1 \in \mathrm{MinNonCPos}(t_1)$ and $p_2 \in \mathrm{MinNonCPos}(t_2)$, $t_1/p_1 = t_2/p_2$ iff $\hat{t}_1/p_1 = \hat{t}_2/p_2$
  - $|\hat{t}_1| < |\hat{t}_2|$ and $|\hat{t}_1|_x \leq |\hat{t}_2|_x$ for every $x \in V^{\mathrm{C}}$
  - $\mathrm{def}(u_1)\,, \ldots,\ \mathrm{def}(u_k)$ are exactly those definedness atoms in

    $$\{\, \mathrm{def}(t_i/p) \mid i \in \{1, 2\} \wedge p \in \mathrm{MinNonCPos}(t_i) \,\}$$

    that do not occur in $\overline{\Gamma}, \overline{\Delta}$.

Removing Redundant Literals

Multiple Literals:
`mult-lit` *m n*

$$\frac{\langle \Gamma, \lambda, \Delta, \lambda', \Pi \,;\, w\rangle}{\langle \Gamma, \lambda, \Delta, \Pi \,;\, w\rangle}\qquad \text{if} \quad \begin{array}{l} - \ (\Gamma, \lambda, \Delta, \lambda', \Pi)[m] = \lambda \\ - \ (\Gamma, \lambda, \Delta, \lambda', \Pi)[n] = \lambda' \\ - \ \lambda' =_{\text{lit}} \lambda. \end{array}$$

$=$-Removal:
`=-removal` *m*

$$\frac{\langle \Gamma,\ t_1 = t_2\,,\Delta\,;\ w\rangle}{\langle \Gamma, \Delta\,;\ w\rangle}$$

if
- $(\Gamma,\ t_1 = t_2\,, \Delta)[m] =\ t_1 = t_2$
- $t_1, t_2 \in \mathcal{T}(\mathsf{sig}^C, V^{\mathrm{C}})$
- t_1 and t_2 are not unifiable.

$<$-Removal:
`<-removal` *m*

$$\frac{\langle \Gamma,\ (u_1,\ldots,u_k) < ()\,,\Delta\,;\ w\rangle}{\langle \Gamma, \Delta\,;\ w\rangle}$$

if $(\Gamma,\ (u_1,\ldots,u_k) < ()\,, \Delta)[m] =$ $(u_1,\ldots,u_k) < ()\,.$

$\neq$-Removal:
`=/=-removal` *m*

$$\frac{\langle \Gamma,\ t_1 \neq t_2\,,\Delta\,;\ w\rangle}{\langle \Gamma, \Delta\,;\ w\rangle}$$

if
- $(\Gamma,\ t_1 \neq t_2\,, \Delta)[m] =\ t_1 \neq t_2$
- $\mathrm{top}(t_1) = \mathrm{top}(t_2)$
- each atom in $\{\, t_1/p \neq t_2/p \mid p \in \mathrm{MinDifPos}(t_1,t_2)\,\}$ occurs in Γ, Δ.

$\neg\mathrm{def}$-Removal:
`~def-removal` *m*

$$\frac{\langle \Gamma,\ \neg\mathrm{def}(t)\,,\Delta\,;\ w\rangle}{\langle \Gamma, \Delta\,;\ w\rangle}$$

if
- $(\Gamma,\ \neg\mathrm{def}(t)\,, \Delta)[m] =\ \neg\mathrm{def}(t)$
- $\mathrm{top}(t) \in (C \cup V^{\mathrm{C}})$
- each atom in $\{\, \neg\mathrm{def}(t/p) \mid p \in \mathrm{MinNonCPos}(t)\,\}$ occurs in Γ, Δ.

Making Use of Negative Literals

Constant Rewriting:
`const-rewrite` *m n p*

$$\frac{\langle \Gamma, \lambda[t_1]_p, \Delta\,;\ w\rangle}{\langle \Gamma, \lambda[t_2]_p, \Delta\,;\ w\rangle}$$

if
- $(\Gamma, \lambda, \Delta)[m] = \lambda$
- $p \in \mathrm{Pos}(\lambda)$ and $\lambda/p = t_1$
- there is a literal $t_1 \dot{\neq} t_2$ in Γ, Δ such that $(\Gamma, \lambda, \Delta)[n] =\ t_1 \dot{\neq} t_2\,.$

$\neq$-Unification:
`=/=-unif` *m*

$$\frac{\langle \Gamma,\ t_1 \neq t_2\,,\Delta\,;\ w\rangle}{\langle \Gamma\tau, \Delta\tau\,;\ w\tau\rangle}$$

if
- $(\Gamma,\ t_1 \neq t_2\,, \Delta)[m] =\ t_1 \neq t_2$
- $t_1, t_2 \in \mathcal{T}(\mathsf{sig}^C, V^{\mathrm{C}})$
- $\tau = \mathrm{mgu}(\,t_1 = t_2\,, \mathrm{Var}(\Gamma, t_1, t_2, \Delta, w))$ exists.

Constructor Variable Addition:
`ctr-var-add` *m x*

$$\frac{\langle \Gamma,\ \neg\mathrm{def}(t)\,,\Delta\,;\ w\rangle}{\langle \Gamma,\ x \neq t\,, \Delta\,;\ w\rangle}$$

if
- $(\Gamma,\ \neg\mathrm{def}(t)\,, \Delta)[m] =\ \neg\mathrm{def}(t)$
- $x \in V^{\mathrm{C}} \backslash \mathrm{Var}(\Gamma, \Delta, t, w).$

Further Inference Rules for Order Atoms

Tuple $<$-Reduction:
`tuple <-reduct` *k*

$$\frac{\langle \Gamma,\ (t_1, u_1, \ldots, u_m) < (t_2, v_1, \ldots, v_n)\,, \Delta\,;\ w\rangle}{\langle\, t_1 < t_2\,, \Gamma,\ (t_1, u_1, \ldots, u_m) < (t_2, v_1, \ldots, v_n)\,, \Delta\,;\ w\rangle}$$

if
- $(\Gamma, (t_1, u_1, \ldots, u_m) < (t_2, v_1, \ldots, v_n)\,, \Delta)[k] =$ $(t_1, u_1, \ldots, u_m) < (t_2, v_1, \ldots, v_n)$
- $m > 0\ \vee\ n > 0.$

Tuple $=$-Reduction:
`tuple =-reduct` *k*

$$\frac{\langle \Gamma,\ (t, u_1, \ldots, u_m) < (t, v_1, \ldots, v_n)\,, \Delta\,;\ w\rangle}{\langle \Gamma,\ (u_1, \ldots, u_m) < (v_1, \ldots, v_n)\,, \Delta\,;\ w\rangle}$$

if $(\Gamma, (t, u_1, \ldots, u_m) < (t, v_1, \ldots, v_n)\,, \Delta)[k] =$ $(t, u_1, \ldots, u_m) < (t, v_1, \ldots, v_n)\,.$

$<$-Monotonicity:
`<-mono` *m* $p_1\ p_2$

$$\frac{\langle \Gamma,\ t_1[u_1]_{p_1} < t_2[u_2]_{p_2}\,,\, \Delta\,;\, w\rangle}{\langle\, u_1 < u_2\,,\, \Gamma,\ t_1[u_1]_{p_1} < t_2[u_2]_{p_2}\,,\, \Delta\,;\, w\rangle}$$

if
- $(\Gamma,\ t_1 < t_2\,,\, \Delta)[m] =\ t_1 < t_2$
- for $i \in \{1,2\}$, $\ p_i \in \mathrm{Pos}(t_i)\backslash\{\varepsilon\}$ and $\ t_i/p_i = u_i$
- there are $\hat{t}_1, \hat{t}_2 \in \mathcal{T}(\mathrm{sig}^C, V^{\mathrm{C}})$ and $x_1, x_2 \in V^{\mathrm{C}}\backslash \mathrm{Var}(t_1, t_2)$ such that
  - for $i \in \{1,2\}$, $\ \hat{t}_i = t_i[x_i]_{p_i}$
  - $|\hat{t}_1| \leq |\hat{t}_2|$
  - for every $x \in V^{\mathrm{C}}\backslash\{x_1, x_2\}$, $\ |\hat{t}_1|_x \leq |\hat{t}_2|_x$.

$<$-Transitivity:
`<-trans` *m* w_2

$$\frac{\langle \Gamma,\ w_1 < w_3\,,\, \Delta\,;\, w\rangle}{\langle\, w_1 < w_2\,,\, \Gamma,\ w_1 < w_3\,,\, \Delta\,;\, w\rangle \ \langle\, w_2 < w_3\,,\, \Gamma,\ w_1 < w_3\,,\, \Delta\,;\, w\rangle}$$

if $\quad (\Gamma,\ w_1 < w_3\,,\, \Delta)[m] =\ w_1 < w_3$.

Non-Applicative Case Analyses

Substitution Addition:
`subst-add` $\sigma_1\ \ldots\ \sigma_n$.

$$\frac{\langle \Gamma\,;\, w\rangle}{\langle \Gamma\sigma_1\,;\, w\sigma_1\rangle\ \ldots\ \langle \Gamma\sigma_n\,;\, w\sigma_n\rangle}$$

if $\{\sigma_1, \ldots, \sigma_n\}$ is a cover set of substitutions for $\langle \Gamma\,;\, w\rangle$.

Literal Addition:
`lit-add` $\lambda_1\ \ldots\ \lambda_n$.

$$\frac{\langle \Gamma\,;\, w\rangle}{\langle \Lambda_1, \Gamma\,;\, w\rangle\ \ldots\ \langle \Lambda_n, \Gamma\,;\, w\rangle\ \langle \Lambda, \Gamma\,;\, w\rangle}$$

if $\Lambda_1, \ldots, \Lambda_n, \Lambda$ is the case analysis resulting from literals $\lambda_1, \ldots, \lambda_n$ for $n > 0$.

Applicative Inference Rules

Non-Inductive Applicative Inference Rules

Non-Inductive Subsumption:
`axiom-subs` *DefRule* μ

$$\frac{\langle \Gamma\,;\, w\rangle}{\langle \Lambda_1, \Gamma\,;\, w\rangle\ \ldots\ \langle \Lambda_n, \Gamma\,;\, w\rangle} \quad \text{with } \langle \Pi\,;\, ()\rangle^{\mathcal{L}}$$

if there is a substitution μ and a clause Θ such that
- the clause representation of the defining rule *DefRule* is Π
- Γ, Θ contains $\mathrm{DefCond}(\mu, \Pi), \Pi\mu$
- $\Lambda_1, \ldots, \Lambda_n, \Lambda$ is the case analysis resulting from Θ.

`lemma-subs` *PS-Tree* μ

$$\frac{\langle \Gamma\,;\, w\rangle}{\langle \Lambda_1, \Gamma\,;\, w\rangle\ \ldots\ \langle \Lambda_n, \Gamma\,;\, w\rangle} \quad \text{with } \langle \Pi\,;\, \hat{w}\rangle^{\mathcal{L}}$$

if there is a substitution μ and a clause Θ such that
- the goal labeling the root of the proof state tree *PS-Tree* is $\langle \Pi\,;\, \hat{w}\rangle$
- Γ, Θ contains $\mathrm{DefCond}(\mu, \Pi), \Pi\mu$
- $\Lambda_1, \ldots, \Lambda_n, \Lambda$ is the case analysis resulting from Θ.

Non-Inductive Rewriting:
`axiom-rewrite` *m p DefRule n* μ

$$\frac{\langle \Gamma, \lambda, \Delta\,;\, w\rangle}{\langle \Lambda_1, \Gamma, \lambda, \Delta\,;\, w\rangle\ \ldots\ \langle \Lambda_n, \Gamma, \lambda, \Delta\,;\, w\rangle\ \langle \Lambda, \Gamma, \lambda[r\mu]_p, \Delta\,;\, w\rangle}$$

with $\langle \Pi,\ l \doteq r\,,\, \Sigma\,;\, ()\rangle^{\mathcal{L}}$

if there is a position $p \in \mathrm{Pos}(\lambda)$, a substitution μ and a clause Θ such that
- $(\Gamma, \lambda, \Delta)[m] = \lambda$
- the clause representation of the defining rule *DefRule* is $\Pi,\ l \doteq r\,,\, \Sigma$
- $(\Pi,\ l \doteq r\,,\, \Sigma)[n] =\ l \doteq r$
- $\lambda/p = l\mu$
- $\Gamma, \Delta,\ l\mu = r\mu\,,\, \Theta$ contains $\mathrm{DefCond}(\mu, (\Pi,\ l \doteq r\,,\, \Sigma)), \Pi\mu, \Sigma\mu$
- $\Lambda_1, \ldots, \Lambda_n, \Lambda$ is the case analysis resulting from Θ.

`lemma-rewrite` *m p PS-Tree n* μ

$$\frac{\langle \Gamma, \lambda, \Delta \,;\, w\rangle}{\langle \Lambda_1, \Gamma, \lambda, \Delta \,;\, w\rangle \ \ldots\ \langle \Lambda_n, \Gamma, \lambda, \Delta \,;\, w\rangle\ \langle \Lambda, \Gamma, \lambda[r\mu]_p, \Delta \,;\, w\rangle}$$

with $\langle \Pi,\, l \doteq r\,, \Sigma \,;\, \hat{w}\rangle^{\mathcal{L}}$

if there is a position $p \in \mathrm{Pos}(\lambda)$, a substitution μ and a clause Θ such that
- $(\Gamma, \lambda, \Delta)[m] = \lambda$
- the goal labeling the root of the proof state tree *PS-Tree* is $\langle \Pi,\, l \doteq r\,, \Sigma \,;\, \hat{w}\rangle$
- $(\Pi,\, l \doteq r\,, \Sigma)[n] = \ l \doteq r$
- $\lambda/p = l\mu$
- $\Gamma, \Delta,\, l\mu = r\mu\,, \Theta$ contains $\mathrm{DefCond}(\mu, (\Pi,\, l \doteq r\,, \Sigma)), \Pi\mu, \Sigma\mu$
- $\Lambda_1, \ldots, \Lambda_n, \Lambda$ is the case analysis resulting from Θ.

Applicative Literal Removal:
`appl-lit-removal` *m* μ *Lma n*

$$\frac{\langle \Gamma, \lambda, \Delta \,;\, w\rangle}{\langle \Gamma, \Delta \,;\, w\rangle} \quad \text{with } \langle \Pi, \lambda', \Sigma \,;\, \hat{w}\rangle^{\mathcal{L}}$$

if there is a substitution μ such that
- $(\Gamma, \lambda, \Delta)[m] = \lambda$
- if Lma is a defining rule then Π, λ', Σ is the clause representation of Lma
- if Lma is a proof state tree then the root of Lma is labeled with $\langle \Pi, \lambda', \Sigma \,;\, \hat{w}\rangle$
- $(\Pi, \lambda', \Sigma)[n] = \lambda'$
- $\lambda =_{\mathrm{lit}} \overline{\lambda'\mu}$
- Γ, Δ contains $\mathrm{DefCond}(\mu, (\Pi, \lambda', \Sigma)), \Pi\mu, \Sigma\mu$.

Inductive Applicative Inference Rules

Inductive Subsumption:
`ind-subs` *PS-Tree* μ

$$\frac{\langle \Gamma \,;\, w\rangle}{\langle \Lambda_1, \Gamma \,;\, w\rangle \ \ldots\ \langle \Lambda_n, \Gamma \,;\, w\rangle\ \langle\, \hat{w}\mu < w\,, \Lambda, \Gamma \,;\, w\rangle} \quad \text{with } \langle \Pi \,;\, \hat{w}\rangle^{\mathcal{I}}$$

if there is a substitution μ and a clause Θ such that
- the goal labeling the root of the proof state tree *PS-Tree* is $\langle \Pi \,;\, \hat{w}\rangle$
- Γ, Θ contains $\mathrm{DefCond}(\mu, \Pi), \Pi\mu$
- $\Lambda_1, \ldots, \Lambda_n, \Lambda$ is the case analysis resulting from Θ.

Inductive Rewriting:
`ind-rewrite` *m p PS-Tree n* μ

$$\frac{\langle \Gamma, \lambda, \Delta \,;\, w\rangle}{\langle \Lambda_1, \Gamma, \lambda, \Delta \,;\, w\rangle \ \ldots\ \langle \Lambda_n, \Gamma, \lambda, \Delta \,;\, w\rangle\ \langle \Lambda, \Gamma, \lambda[r\mu]_p, \Delta \,;\, w\rangle\ \langle\, \hat{w}\mu < w\,, \Lambda', \Gamma, \lambda, \Delta \,;\, w\rangle}$$

with $\langle \Pi,\, l \doteq r\,, \Sigma \,;\, \hat{w}\rangle^{\mathcal{I}}$

if there is a position $p \in \mathrm{Pos}(\lambda)$, a substitution μ and a clause Θ such that
- $(\Gamma, \lambda, \Delta)[m] = \lambda$
- the goal labeling the root of the proof state tree *PS-Tree* is $\langle \Pi,\, l \doteq r\,, \Sigma \,;\, \hat{w}\rangle$
- $(\Pi,\, l \doteq r\,, \Sigma)[n] = \ l \doteq r$
- $\lambda/p = l\mu$
- $\Gamma, \Delta,\, l\mu = r\mu\,, \Theta$ contains $\mathrm{DefCond}(\mu, (\Pi,\, l \doteq r\,, \Sigma)), \Pi\mu, \Sigma\mu$
- $\Lambda_1, \ldots, \Lambda_n, \Lambda$ is the case analysis resulting from Θ
- $\Lambda' = \ l\mu = r\mu\,, \Lambda$.

Zur Korrektheit und Sicherheit von Inferenzregeln (1)

Beachte

Alle Inferenzregeln haben die Form

$$\frac{\langle \Gamma; \omega \rangle}{\langle \Gamma_1; \omega_1 \rangle \cdots \langle \Gamma; \omega_n \rangle} \quad (\text{mit } \langle \Pi; \omega' \rangle^{\mathcal{U}})$$

Dabei ist entweder $\omega_i \equiv \omega$ für alle i oder $\omega_i \equiv \sigma_i(\omega)$ für alle i.

Die Regel ist genau dann nicht-applikativ, falls der Teil (mit $\langle \Pi; \omega' \rangle^{\mathcal{U}}$) entfällt.

Beachte

Gilt $\Gamma \subseteq \Gamma_i$ für alle i, so ist die Regel sicher.

Frage

Wie zeigt man, dass die Regel korrekt ist?

Zur Korrektheit und Sicherheit von Inferenzregeln (2)

Beachte

a) Sei die Inferenzregel nicht-applikativ oder Π gültig. Die Inferenzregel ist korrekt, falls für jedes $\mathcal{A} \in$ DMod(spec) gilt

$$\mathcal{A} \models \Gamma \text{ , falls } \mathcal{A} \models \Gamma_i \text{ für alle } i$$

In diesem Fall bleibt die Größe der Gegenbeispiele erhalten.
Die Inferenzregel ist sicher, falls für jedes $\mathcal{A} \in$ DMod(spec) gilt

$$\mathcal{A} \models \Gamma_i \text{ für alle } i \text{ , falls } \mathcal{A} \models \Gamma$$

b) Sei die Inferenzregel applikativ, $\mathcal{U} = \mathcal{I}$ und Π nicht gültig.
Die Inferenzregel ist korrekt, falls für alle $\mathcal{A} \in$ DMod(spec) und jedes $\mathcal{A}$-Gegenbeispiel $(\langle \Gamma; \omega \rangle, \sigma)$ ein $\mathcal{A}$-Gegenbeispiel $(\langle \Pi; \omega' \rangle, \tau)$ gibt mit $(\langle \Pi; \omega' \rangle, \tau) <_{\mathcal{A}} (\langle \Gamma; \omega \rangle, \sigma)$.
In diesem Fall werden Gegenbeispiele durch kleinere ersetzt.

c) Sei Var$(\Gamma, \Gamma_1, \ldots, \Gamma_n) \subseteq V^C$. Es gilt
$\mathcal{A} \models \Gamma \text{ gdw } \mathcal{A} \models \tau(\Gamma)$ für jede induktive Substitution τ.
$\mathcal{A} \models \tau(\Gamma) \text{ gdw } \mathcal{A} \models \tau(\lambda)$ für ein $\lambda \in \Gamma$.

Zur Korrektheit und Sicherheit von Inferenzregeln (3)

Beispiel

Betrachte die Regel

$$\frac{\langle \Gamma_1, \lambda'[t_1]_p, \Gamma_2; \omega\rangle}{\langle \Gamma_1, \lambda'[t_2]_p, \Gamma_2; \omega\rangle} \quad \text{falls } t_1 \neq t_2 \text{ in } \Gamma_1 \cup \Gamma_2$$

Sei $\mathcal{A} \in$ DMod(spec) und $\Gamma = \Gamma_1 \cup \{\lambda'[t_1]_p\} \cup \Gamma_2$ und $\Gamma' = \Gamma_1 \cup \{\lambda'[t_2]_p\} \cup \Gamma_2$ und τ eine induktive Substitution. Gilt $\mathcal{A} \models \tau(\lambda)$ für ein $\lambda \in \Gamma_1 \cup \Gamma_2$, so gilt $\mathcal{A} \models \tau(\Gamma)$ und $\mathcal{A} \models \tau(\Gamma')$.

Sei $\mathcal{A} \not\models \tau(\lambda)$ für alle $\lambda \in \Gamma_1 \cup \Gamma_2$. Dann gilt $\mathcal{A} \models \tau(t_1) = \tau(t_2)$, also gilt $\mathcal{A} \models \tau(\lambda'[t_1]_p) \text{ gdw } \mathcal{A} \models \tau(\lambda'[t_2]_p)$.

Dies zeigt $\mathcal{A} \models \Gamma \text{ gdw } \mathcal{A} \models \Gamma'$.

Also ist die Regel korrekt und sicher.

Eigenschaften der QUODLIBET-Regeln

Satz 4.5.

Die Inferenzregeln von QUODLIBET sind korrekt und sicher.

Beweis:

Übung. Man beachte dazu die Vorüberlegungen.

Bemerkungen zu den Inferenzregeln

1. Bemerkung

Die Anwendung der Inferenzregeln ist hochgradig nicht-deterministisch.

(1) Es sind in der Regel mehrere Inferenzregeln auf ein Ziel $\langle \Gamma; \omega \rangle$ anwendbar.

(2) Es ist i. Allg. eine feste Inferenzregel auf mehrere $\lambda \in \Gamma$ anwendbar, manchmal sogar mit unterschiedlichen Lemmata bzw. Induktions-Hypothesen.

Zu (2)

Die Inferenzregeln sind so geschrieben, dass es sinnvoll ist, eine Regel jeweils auf das erste λ aus (der Liste) Γ anzuwenden.

Zu (1)

In QUODLIBET sind Folgen von Inferenzregeln zu **Taktiken** zusammengefasst.

4.5 Die Beweisführung

Sei

sig $= (S, C, F, \alpha)$ eine Signatur,
spec $=$ (sig, R) eine zulässige Spezifikation,
$\mathcal{L}$ eine Menge von (gültigen) Lemmata, $R \subseteq \mathcal{L}$,
Γ eine Klausel.

Zu zeigen sei DMod(spec) $\models \Gamma$.

Bemerkung

Ein Beweisversuch zu Γ wird als Baum organisiert.

Ein Beweisbaum hat als Knoten Ziel- und Inferenzknoten.

Ein Zielknoten trägt als Marke das Ziel (eine Klausel und ein Gewicht). Er wird durch Anwendung einer Inferenzregel abgearbeitet. Dies erzeugt einen Inferenzknoten mit dem Namen der anzuwendenden Inferenz.

Ein Inferenzknoten trägt als Marke den Namen einer Inferenzregel. Wird die Anwendung der Inferenz auf das Ziel abgearbeitet, so entstehen n neue Ziele mit $n \geq 0$.

Der Beweis ist abgeschlossen, wenn alle Ziel- und Inferenzknoten abgearbeitet sind.

Inferenzfolgen können zu Strategien zusammengefasst werden.

Beweisbäume

Bezeichnung

Ein **Beweisbaum** ist ein Und-Oder-Baum.

Zielknoten sind Oder-Knoten. **Zielknoten** tragen ein Teilziel $\langle \Gamma ; \omega \rangle$. Söhne von Zielknoten sind Inferenzknoten.

Ein **Inferenzknoten** ist ein Und-Knoten. Er trägt den Namen der angewendeten Inferenz und Informationen darüber, welche Klausel $\langle \Pi ; \hat{\omega} \rangle$ wie auf seinen Vaterknoten $\langle \Gamma ; \omega \rangle$ angewendet wird. Söhne des Inferenzknotens sind Zielknoten zu den neu entstandenen Teilzielen $\langle \Gamma_i ; \omega_i \rangle$.

Beachte

Die Wahl, welche Inferenz auf $\langle \Gamma ; \omega \rangle$ angewendet wird, ist frei. Daher sind Zielknoten Oder-Knoten.

Ein Inferenzknoten erzeugt neue Zielknoten, die alle abgearbeitet werden müssen, er ist also ein Und-Knoten.

Ein Beweisbaum ist **abgeschlossen**, wenn jedes Blatt ein Inferenzknoten ist. (Dann ist sein Vater abgearbeitet, ohne dass ein neues Teilziel erzeugt wurde.)

Ein abgeschlossener Beweisbaum mit Wurzel $\langle \Gamma ; \omega \rangle$ stellt einen Beweis für die Gültigkeit von Γ dar.

Behandlung von Gewichten

Start eines Beweisversuchs zu Γ

Es wird ein Beweisbaum mit einzigem Knoten $\langle \Gamma ; \omega_0 \rangle$ erzeugt. Dabei ist ω_0 zunächst nur ein Name $W_\Gamma(x_1, \ldots, x_n)$ mit $\{x_1, \ldots, x_n\} = \mathsf{Var}(\Gamma)$.

Instanziierung von Gewichten

Nur wenige Inferenzregeln, z. B. Fallunterscheidung nach Substitutionen, verändern das Gewicht: $\langle \Gamma ; \omega \rangle$ hat Teilziele der Form $\langle \Gamma' ; \sigma(\omega) \rangle$. Die anderen Inferenzregeln erzeugen Teilziele der Form $\langle \Gamma' ; \omega \rangle$.

Solange das Gewicht noch ein Name der Form $W_\Gamma(t_1, \ldots, t_n)$ ist, wird der Name durch σ instanziiert.

Setzen von Gewichten

Irgendwann muss ein Name $W_\Gamma(t_1, \ldots, t_n)$ zu $(t'_1, \ldots, t'_k) \in \mathsf{Term}(F, V)^k$ „geeignet“ konkretisiert werden. Dies macht der Benutzer direkt oder durch den Aufruf einer Taktik.

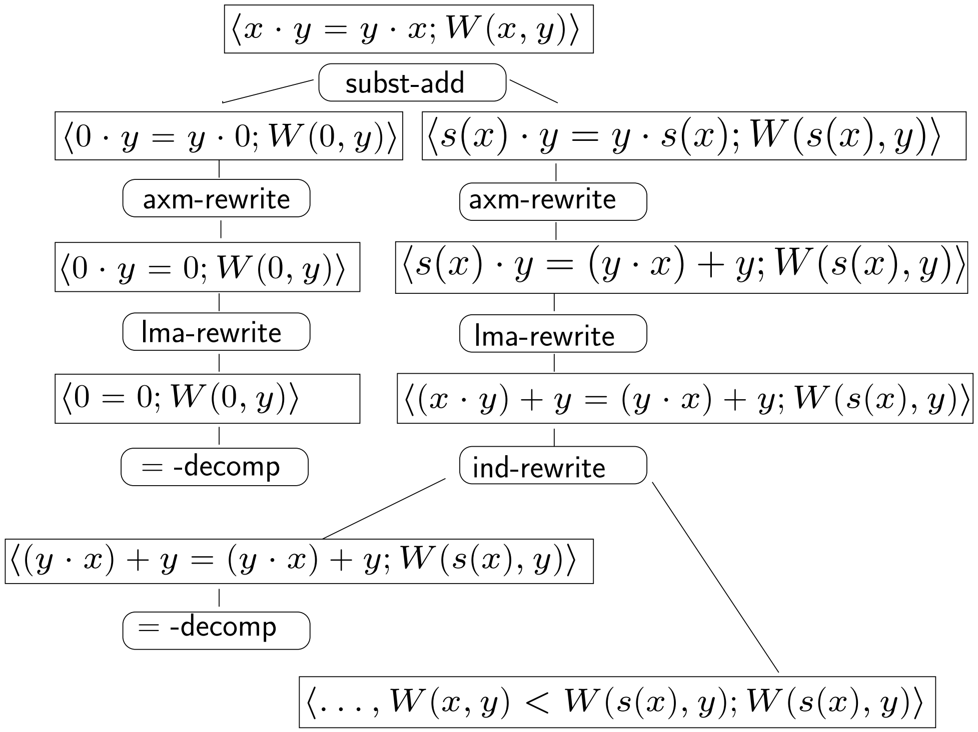
Es entstehen konkrete Beweisaufgaben, die mit den Inferenzregeln abgearbeitet werden müssen.

Beachte

Dies realisiert die Idee, dass die Induktionsordnung erst sehr spät (“bei Bedarf‘) festgelegt wird, und zwar durch das Setzen des Gewichts.

Beispiel: Multiplikation (1)

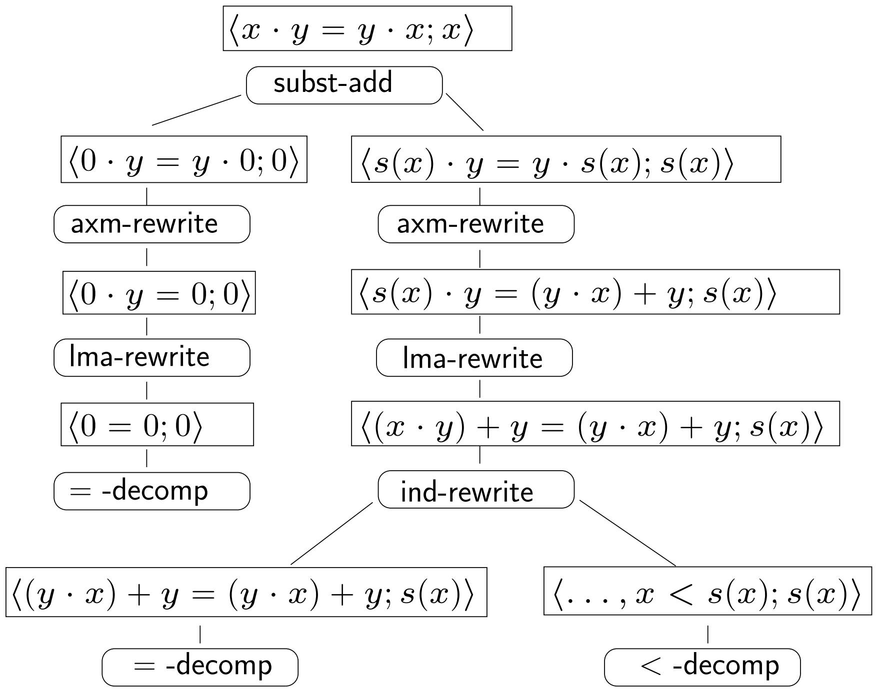
Axiome	$x + 0 = x$	$x + s(y) = s(x + y)$
	$x \cdot 0 = 0$	$x \cdot s(y) = (x \cdot y) + x$
Lemmata	$0 \cdot y = 0$	$s(x) \cdot y = (x \cdot y) + y$
Ziel	$x \cdot y = y \cdot x$	



Beispiel: Multiplikation (2)

Setzen des Gewichtes

$$W(x, y) :\equiv x$$



Proof Engineering

Beobachtung

Die Beweisführung ist nur schwer automatisierbar. Man braucht Kriterien

a) für die Auswahl der „richtigen“ Inferenzregel,
b) für die Definition der Gewichte (dies definiert die Ind.-Ordnung),
c) für die „richtigen“ Lemmata.

Dies bezeichnet man mit **Proof Engineering**.

Ansatz Taktiken entwickeln.

Taktiken zu b)

Dies lässt sich häufig aus der Beweisverpflichtung $W(t_1, \ldots, t_n) < W(t'_1, \ldots, t'_n)$ raten: Als Taktik formulieren.

Taktiken zu c)

Man errate aus dem aktuellen Teilziel $\langle \Gamma; \omega \rangle$ eine Hilfsbehauptung, führe sie mit dem assume-Befehl ein, führe damit den Beweis weiter und beweise sie bei Erfolg.

Vorsicht: (1) Die Hilfsbehauptung kann falsch sein. (2) Dies kann zu unendlichem Prozess führen.

Taktiken zu a)

(1) Inferenzfolgen zusammenfassen

- für die Reinigung von Klauseln,
- für die Termersetzung und Subsumption,
- für die Behandlung von def- und $<$-Literalen.

(2) Analyse der definierten Funktionssymbole f

- dies liefert Hinweise für die Fallunterscheidung nach Substitutionen (Expansion von f),
- dies liefert außerdem das Lemma def$f(x_1, \ldots, x_n)$ und hilft bei dem Beweis von def-Literalen.

(3) Auswahl des „richtigen“ Literals in $\langle \Gamma; \omega \rangle$: Beachte, dass es reicht, **ein** Literal aus Γ zu beweisen

- wähle stets das erste Literal aus Γ, für das eine Inferenzregel anwendbar ist. Die Inferenzregeln sind so entworfen, dass das sinnvoll ist.

Beispiel Multiplikation

Sei

Gegeben: $R: x + 0 = x \qquad x + s(y) = s(x + y)$
$\qquad\qquad x \cdot 0 = 0 \qquad x \cdot s(y) = (x \cdot y) + x$

Ziel: $x \cdot y = y \cdot x$

Man benötigt folgende Hilfsbehauptungen

(1) $0 \cdot y = 0$ (2) $s(x) \cdot y = (x \cdot y) + y$

Es ist (1) leicht, für (2) benötigt man

(3) $((x \cdot y) + y) + x = ((x \cdot y) + x) + y)$

Für (3) rate man die Hilfsbehauptung

(4) $(x + y) + z = x + (y + z)$ (5) $x + y = y + x$

oder

(4') $(z + y) + x = (z + x) + y$ (Generalisierung von (3))

Es gilt

Alle Hilfsbehauptungen lassen sich beweisen.

Damit lässt sich das Ziel beweisen.

Also Hilfsbehauptungen lassen sich häufig beim Beweisen erraten.
$\Rightarrow$ Taktiken dafür entwickeln.

Beispiel Mergesort: Die Spezifikation

merge: Mischen

merge(nil, l_2) $= l_2$ merge(l_1, nil) $= l_1$
merge($x.l_1, y.l_2$) $= x$.merge($l_1, y.l_2$) $\Leftarrow$ leq(x, y) = true
merge($x.l_1, y.l_2$) $= y$.merge($x.l_1, l_2$) $\Leftarrow$ leq(x, y) $\neq$ true,
def leq(x, y)

split1, **split2**: Aufspalten

split1(nil) = nil split2(nil) = nil
split1(x.nil) = x.nil split2(x.nil) = nil
split1($x.y.l$) = x.split1(l) split2($x.y.l$) = y.split2(l)

msort: Sortieren durch Mischen

msort(nil) = nil
msort(x.nil) = x.nil
msort($x.y.l$) = merge(msort(split1($x.y.l$)), msort(split2($x.y.l$)))

sorted: Sortiertheit

sorted(nil) = true
sorted($x.l$) = sorted(l) $\Leftarrow$ leq-list(x, l) = true
sorted($x.l$) = false $\Leftarrow$ leq-list(x, l) $\neq$ true, def leq-list(x, l)

Ziel

sorted(msort(l)) = true.

Mergesort: Die Voraussetzungen

Voraussetzungen

a) Die Sorten BOOL, NAT und LIST mit den üblichen Operationen sind spezifiziert, z. B.
leq(x, y) = true gdw $x \leq y$
leq-list(x, l) = true gdw $x \leq y$ für alle y in der Liste l

b) Es sind hinreichend viele Lemmata bewiesen, z. B.
L-leq-trans$\{$leq(x, y) = true $\wedge$ leq(y, z) = true
$\Rightarrow$ leq(x, z) = true$\}$
L-leq-list-def$\{$def leq-list$(x, l)\}$

c) Alle spezifizierten Operatoren sind analysiert, also sind sinnvolle Fallunterscheidungen nach Substitutionen für ihre Expansion bekannt.

d) Die Totalität aller Operatoren ist bewiesen, z. B.
L-def-merge$\{$def merge$(l_1, l_2)\}$

Beachte

Diese Voraussetzungen sind hier didaktisch für die Vorlesung eingeführt.

In der Praxis muss im Beweisversuch sichergestellt werden, dass sie erfüllt sind.

Beweis zu L-def-merge

1 $\{$def merge$(l_1, l_2); W(l_1, l_2)\}$ subst-add

1.1 $\{$def merge(nil, l_2); W(nil, l_2)$\}$ axm-rewrite
$\{$def l_2; W(nil, l_2)$\}$ $\surd$ mit def-decomp

1.2 $\{$def merge(l_1, nil); W(l_1, nil)$\}$ $\surd$ analog zu 1.1

1.3 $\{$def merge$(x.l_1, y.l_2); W(x.l_1, y.l_2)\}$ lit-add

1.3.1 $\{$leq$(x, y) \neq$ true, def merge$(x.l_1, y.l_2); W(x.l_1, y.l_2)\}$
axm-rewrite
$\{$def x.merge$(l_1, y.l_2), \ldots; W(x.l_1, y.l_2)\}$ def-decomp
$\{$def merge$(l_1, y.l_2), \ldots; W(x.l_1, y.l_2)\}$ ind-subs
$\{W(l_1, y.l_2) < W(x.l_1, y.l_2), \ldots; W(x.l_1, y.l_2)\}$ offen

1.3.2 $\{$leq(x, y) = true, def merge$(x.l_1, y.l_2); W(x.l_1, y.l_2)\}$
axm-rewrite

1.3.2.1 $\{$def leq$(x, y), \ldots; W(x.l_1, y.l_2)\}$ $\surd$ lma-subs

1.3.2.2 $\{$def y.merge$(x.l_1, l_2), \ldots; W(x.l_1, y.l_2)\}$ def-decomp
$\{$def merge$(x.l_1, l_2), \ldots; W(x.l_1, y.l_2)\}$ ind-subs
$\{W(x.l_1, l_2) < W(x.l_1, y.l_2), \ldots; W(x.l_1, l_2)\}$ offen

Setzen des Gewichts $W(l_1, l_2) :\equiv (l_1, l_2)$ liefert

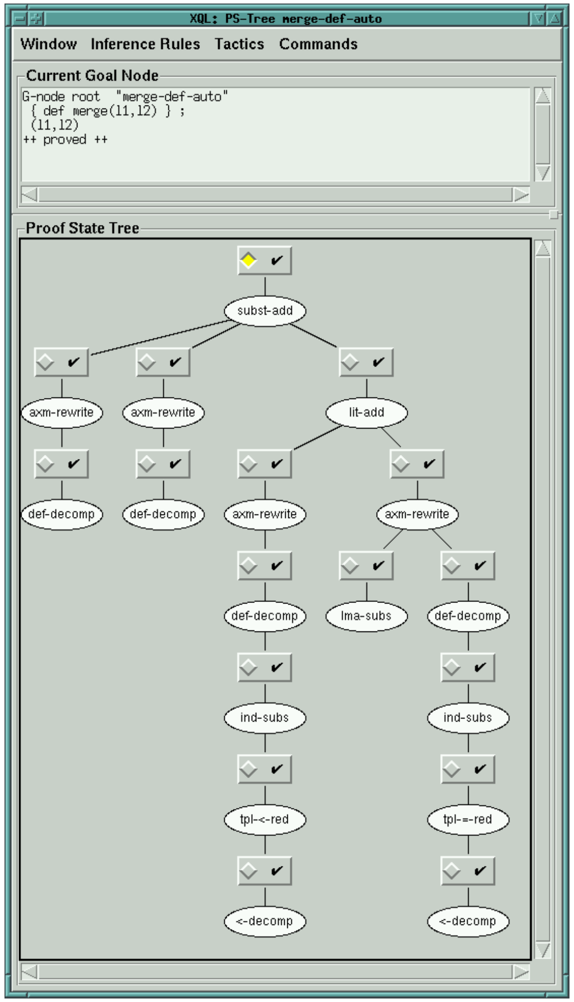
1.3.1 $\{(l_1, y.l_2) < (x.l_1, y.l_2), \ldots; (x.l_1, y.l_2)\}$ $\surd$

1.3.2.2 $\{(x.l_1, l_2) < (x.l_1, y.l_2), \ldots; (x.l_1, y.l_2)\}$ $\surd$

Schreibweise hier

Ein Knoten ist beschrieben durch seine Position im Beweisbaum, sein Teilziel $\langle \Gamma; \omega \rangle$ und die auf ihn angewandte Inferenzregel. Knoten ohne Position sind einzige Söhne ihres Vaters. Knoten mit Markierung $\surd$ sind abgearbeitet ohne ein Teilziel zu erzeugen oder leicht zu zeigen.

Beweiszustandsbaum zu L-def-merge



Proof Engineering

Zentrale Beweisidee zum Beweis von $\{\text{sorted}(\text{msort}(l)) = \text{true}\}$

a) Hilfsbehauptung erzeugen:
L-sorted-merge$\{\text{sorted}(l_1) = \text{true} \wedge \text{sorted}(l_2) = \text{true}$
$\Rightarrow \text{sorted}(\text{merge}(l_1, l_2) = \text{true}\}$

b) Der Beweis hierzu benötigt weitere Hilfsbehauptung
L-leq-merge$\{\text{leq-list}(x, l_1) = \text{true} \wedge \text{leq-list}(x, l_2) = \text{true}$
$\Rightarrow \text{leq-list}(x, \text{merge}(l_1, l_2)) = \text{true}\}$

Induktionsordnung festlegen

Idee aus der Spezifikation von msort:
split1$(x.y.l) < x.y.l$ und split2$(x.l) < x.l$

Also Hilfsbehauptungen
L-split1-ind$\{l \neq \text{nil} \wedge \text{le}(l) \neq s(0) \Rightarrow \text{split1}(l) < l\}$
L-split2-ind$\{l \neq \text{nil} \Rightarrow \text{split2}(l) < l\}$

Also le(l) (=Länge von l) spezifizieren.

Beachte

$<$ ist ein vordefiniertes Prädikatssymbol (nämlich die semantische Ordnung auf Gewichten).

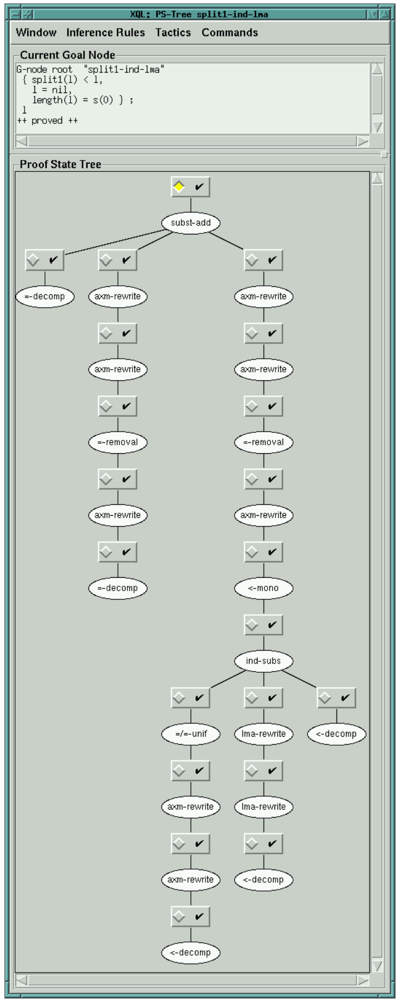
Analog sind $=$ und def vordefinierte Prädikatssymbole.

Beweis zu L-split1-ind

Hilfslemma: L-le1-split1$\{\text{le}(l) = s(0) \Rightarrow \text{split1}(l) = l\}$

1	$\{l = \text{nil}, \text{le}(l) = s(0), \text{split1}(l) < l; W(l)\}$	subst-add
1.1	$\{\text{nil} = \text{nil}, \ldots; W(\text{nil})\}$ $\surd$	mit =-decomp
1.2	$\{x.\text{nil} = \text{nil}, \text{le}(x.\text{nil}) = s(0), \ldots; W(x.\text{nil})\}$	axm-rewrite
	$\{\ldots, s(0) = s(0), \ldots; W(x.\text{nil})\}$ $\surd$	mit =-decomp
1.3	$\{\text{split1}(x.y.l) < x.y.l, \ldots; W(x.y.l)\}$	axm-rewrite
	$\{x.\text{split1}(l) < x.y.l, \ldots; W(x.y.l)\}$	<-mono
	$\{\text{split1}(l) < l, x.\text{split1}(l) < x.y.l, \ldots; W(x.y.l)\}$	ind-subs
1.3.1	$\{l \neq \text{nil}, \ldots, x.\text{split1}(l) < x.y.l, \ldots; W(x.y.l)\}$	$\neq$-unif
	$\{x.\text{split1}(\text{nil}) < x.y.\text{nil}, \ldots; W(x.y.l)\}$	axm-rewrite
	$\{x.\text{nil} < x.y.\text{nil}, \ldots; W(x.y.l)\}$ $\surd$	mit <-decomp
1.3.2	$\{\text{le}(l) \neq s(0), x.\text{split1}(l) < x.y.l, \ldots; W(x.y.l)\}$	lma-rewrite
	$\{\text{le}(l) \neq s(0); x.l < x.y.l, \ldots; W(x.y.l)\}$ $\surd$	<-decomp
1.3.3	$\{W(l) < W(x.y.l), \ldots; W(x.y.l)\}$	Gewicht setzen
	$\{l < x.y.l, \ldots; x.y.l\}$ $\surd$	mit <-decomp

Beweiszustandsbaum zu L-split1-ind



Beweis des Ziels (1)

1	$\{\text{sorted}(\text{msort}(l)) = \text{true}; W(l)\}$	subst-add
1.1	$\{\text{sorted}(\text{msort}(\text{nil})) = \text{true}; W(\text{nil})\}$ $\surd$	
1.2	$\{\text{sorted}(\text{msort}(x.\text{nil})) = \text{true}; W(x.\text{nil})\}$ $\surd$	
1.3	$\{\text{sorted}(\text{msort}(x.y.l)) = \text{true}; W(x.y.l)\}$	axm-rewrite
	$\{\text{sorted}(\text{merge}(\text{msort}(x.\text{split1}(l)), \text{msort}(y.\text{split2}(l)))) = \text{true}; W(x.y.l)\}$	
1.3.1	$\{\text{def msort}(x.\text{split1}(l)), \ldots; W(x.y.l)\}$ $\surd$	
1.3.2	$\{\text{def msort}(y.\text{split2}(l)), \ldots; W(x.y.l)\}$ $\surd$	
1.3.3	$\{\text{sorted }(\text{msort}(x.\text{split1}(l)) = \text{true}, \ldots; W(x.y.l)\}$	ind-rewrite
1.3.3.1	$\{\text{def } x.\text{split1}(l), \ldots; W(x.y.l)\}$ $\surd$	
1.3.3.2	$\{\text{true} = \text{true}, \ldots; W(x.y.l)\}$ $\surd$	mit =-decomp
1.3.3.3	$\{W(x.\text{split1}(l)) < W(x.y.l), \ldots; W(x.y.l)\}$	offen
1.3.4	$\{\text{sorted}(\text{msort}(y.\text{split2}(l))) = \text{true}, ..; W(x.y.l)\}$	ind-rewrite
1.3.4.1	$\{\text{def } y.\text{split2}(l), \ldots; W(x.y.l)\}$ $\surd$	
1.3.4.2	$\{\text{true} = \text{true}, \ldots; W(x.y.l)\}$ $\surd$	mit =-decomp
1.3.4.3	$\{W(y.\text{split2}(l)) < W(x.y.l), \ldots; W(x.y.l)\}$	offen
1.3.5	$\{\text{true} = \text{true}, \ldots; W(x.y.l)\}$ $\surd$	mit =-decomp

Beweis des Ziels (2)

Setzen des Gewichts $W(l) :\equiv l$ liefert

1.3.3.3

$\{x.\text{split1}(l) < x.y.l, \ldots; x.y.l\}$	<-mono
$\{\text{split1}(l) < l, x.\text{split1}(l) < x.y.l, \ldots; x.y.l\}$	lma-subs

1.3.3.3.1

$\{l \neq \text{nil}, \ldots, x.\text{split1}(l) < x.y.l, \ldots; x.y.l\}$	$\neq$-unif
$\{\ldots, x.\text{nil} < x.y.\text{nil}, \ldots; x.y.l\}$ $\surd$	mit <-decomp

1.3.3.3.2

$\{\text{le}(l) \neq s(0), x.\text{split1}(l) < x.y.l, \ldots; x.y.l\}$	lma-rewrite
$\{x.l < x.y.l, \ldots; x.y.l\}$ $\surd$	mit <-decomp

1.3.4.3

$\{y.\text{split2}(l) < x.y.l, \ldots; x.y.l\}$	<-mono
$\{\text{split2}(l) < l, y.\text{split2}(l) < x.y.l, \ldots; x.y.l\}$	lma-subs
$\{l \neq \text{nil}, \ldots, y.\text{split2}(l) < x.y.l, \ldots; x.y.l\}$	$\neq$-unif
$\{y.\text{nil} < x.y.\text{nil}, \ldots; x.y.l\}$ $\surd$	mit <-decomp

Beachte

Die angewandten Lemmata sind

L-sorted-merge bei 1.3
L-split1-ind bei 1.3.3.3
L-split2-ind bei 1.3.4.3
L-le1-split1 bei 1.3.3.3.2

Literatur

Avenhaus, J.; Madlener, K.:

Theorem Proving in Hierarchical Clausal Specifications, in Du, Ko (eds.): Advances in Algorithms, Languages, and Complexity, Kluwer 1997, pp. 1-51.

Bachmair, L.:

Proof by consistency in equational theories, in Proc. Third Annual IEEE Symposium on Logic in Computer Science, 1988, pp. 228-233.

Ehrig, H.; Mahr, B.:

Fundamentals of Algebraic Specification, Springer 1985.

Giesl, J.; Kapur, D.:

Decidable Classes of Inductive Theorems, in Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2001), Lecture Notes in Artificial Intelligence 2083, Springer 2001, pp. 469-484.

Giesl, J.:

Induction Proofs with Partial Functions, in Journal of Automated Reasoning, 2001, Vol. 26, No. 1, pp. 1-49.

Jouannaud, J. P.; Kounalis, E.:

Automatic proofs by induction in equational theories without constructors, in Proc. of the Annual IEEE Symposium on Logic in Computer Science, 1986, pp. 358-366.

Kapur, D.; Subramaniam, M.:

New uses of linear arithmetic in automated theorem proving by induction, in Journal of Automated Reasoning, 1996, Vol. 16, pp. 39-78.

Kühler, U.:

A tactic based inductive theorem prover for data types with partial operations, in DISKI, Dissertationen zur künstlichen Intelligenz, 238, 1999.

Walther, C.:

Mathematical induction, in C. J. Hogger D. M. Gabbay and J. A. Robinson (Eds.): Handbook of Logic in Artificial Intelligence and Logic Programming, Oxford University Press 1994, Vol. 2, pp. 122-227.

Zhang, H.; Kapur, D.; Krishnamoorthy, M. S.:

A Mechanizable Induction Principle for Equational Specifications, in E. Lush and R. Overbeek (Eds): Proc. 9th International Conference on Automated Induction, Lecture Notes in Computer Science 310, Springer 1988, pp. 162-181.