Constraintbasierte Vervollständigungstechniken: Ordnungsconstraints

René Rondot

Dezember 2003

Projektarbeit

Technische Universität Kaiserslautern

Betreuer:
Prof. Dr. Jürgen Avenhaus
Dipl.-Inform. Bernd Löchner

Inhaltsverzeichnis

0. Motivation und Zielsetzung

Schon seit einiger Zeit werden für das automatische Theorembeweisen sogenannte Vervollständigungstechniken eingesetzt. Mit diesen ist es möglich, das Ziel des automatischen Theorembeweisens in naheliegender Weise zu erreichen. Dieses Ziel ist es, zu zeigen, dass aus einer gegebenen Menge von Axiomen eine bestimmte Konklusion folgt. Dass eine Konklusion aus einer Menge von Axiomen folgt, ist dabei genau dann der Fall, wenn alle Modelle der Axiome auch die Konklusion erfüllen. Sowohl die Axiome als auch die Konklusion sind hier Hornklauseln der Prädikatenlogik erster Stufe, bei der als einziges Prädikat die Gleichheit zugelassen ist. Vervollständigung bedeutet nun, dass zu der ursprünglichen Menge von Axiomen sukzessive Folgerungen hinzugenommen werden. Dies wird solange fortgesetzt, bis die zu zeigende Konklusion selbst gefolgert werden kann.

Es gibt verschiedene Möglichkeiten, die Folgerungen zu bestimmen, die hinzugenommen werden sollen. Eine davon, die hier untersucht werden soll, ist die sogenannte Superposition. Deren Grundprinzip ist es, Gleiches durch Gleiches zu ersetzen. Bei der Superposition besteht jedoch das große Problem, dass die Zahl der neu erzeugten Klauseln im Allgemeinen sehr groß ist. Daher ist es wichtig, die erzeugten Klauseln so einzuschränken, dass möglichst schnell das Ziel erreicht wird, ohne zwischendurch zu viele nutzlose Klauseln zu erzeugen. Dies kann durch vielfältige Techniken erreicht werden. Viele dieser Techniken wurden bereits ausführlich untersucht und sind auch in die Praxis umgesetzt worden. Eine grundsätzliche Einschränkung kann durch Einsatz bestimmter Ordnungen erzielt werden, mit deren Hilfe man die Erzeugung von Klauseln, die keine Bedeutung bei der Folgerung der Konklusion haben, teilweise verhindert. Dies geschieht, indem Ordnungsbedingungen aufgestellt werden, unter denen Klauseln erzeugt werden dürfen. Außerdem wurden Redundanzkriterien entwickelt, mit denen ebenfalls die Erzeugung von Klauseln eingeschränkt werden kann oder sogar bereits erzeugte Klauseln wieder entfernt werden können.

Anfang der neunziger Jahre kam eine neue Idee auf, von der eine deutliche Einschränkung erhofft wurde, die sogenannten Constraints (siehe [KKR90]). Constraints dienen dazu, Klauseln, die Variablen enthalten, hinsichtlich Ihrer Instantiierung einzuschränken. Dazu gibt es zwei verschiedene Arten von Constraints: Gleichheitsconstraints und Ordnungsconstraints. Die Gleichheitsconstraints entstehen dabei aus dem Prinzip der Superposition, Gleiches durch Gleiches zu ersetzten. Sie dienen dazu, Positionen in den Klauseln zu markieren, an denen die Superposition nicht mehr durchgeführt werden muss, und schränken dadurch die Anzahl der erzeugten Klauseln ein. Ordnungsconstraints hingegeben entstehen aus den zuvor genannten Ordnungsbedingungen und realisieren zunächst genau die Einschränkungen, die durch die Ordnungsbedingungen entstehen. Vererbt man jedoch die Constraints einer Klausel auf die aus ihr erzeugten Klauseln, so ergibt sich die Möglichkeit, die Ordnungsbedingungen mehrerer Superpositionsschritte in ihrer Gesamtheit auf Erfüllbarkeit zu prüfen. Dies ist insbesondere interessant, weil die Klauseln Variablen enthalten, deren Belegung nicht bekannt ist, und auf denen Ordnungsbedingungen somit nicht unmittelbar überprüft werden können. Ergibt sich dann im Laufe einer Vervollständigung beispielsweise zunächst die Ordnungsbedingung $x > y$, welche für sich erfüllbar ist, und im späteren Verlauf die Bedingung $y > x$, welche ebenfalls im Allgemeinen erfüllbar ist, so kann man jedoch feststellen, dass beide gemeinsam nicht erfüllbar sind und muss dann die betreffende Klausel nicht weiter beachten.

Leider bringt die Verwendung von Constraints auch Nachteile. Zum einen ist die Verwaltung von Klauseln mit Constraints deutlich aufwändiger und außerdem lassen sich einige der sonst verwendeten Redundanzkriterien nicht mehr unverändert anwenden. Die von der Superposition ohne Constraints bekannten Redundanzkriterien für die Verwendung mit Constraints anzupassen, ist ein Ziel dieser Arbeit. Als Basis dafür dient die Arbeit von Bachmair et al. [BGLS95]. Ein weiteres Ziel ist es, die Ideen in einer konkreten Implementierung umzusetzen und festzustellen, inwieweit der Einsatz von Constraints Vorteile bringen kann. Dabei wird insbesondere die Verwendung von Ordnungsconstraints untersucht, über die bisher leider nur sehr wenige Erkenntnisse vorliegen. Genauere Untersuchungen hinsichtlich Gleichheitsconstraints finden sich in einer Arbeit von Thomas Türk [Tür03]. Neben [BGLS95] ist auch [GN01] Grundlage dieser Arbeit.

Im ersten Abschnitt sollen zunächst grundlegende Definitionen festgelegt werden, die im folgenden Text benötigt werden. Der verwendete Superpositionskalkül für Hornklauseln wird dann im zweiten Abschnitt genauer beschrieben und dessen Vollständigkeit bewiesen. Danach wird die generelle Verwendung von Constraints im dritten Abschnitt erläutert und der vorgestellte Superpositionskalkül dementsprechend erweitert. Im vierten Abschnitt wird dann erläutert, wie sich mittels Redundanzkriterien weitere Einschränkungen treffen lassen. Konkrete Redundanzkriterien werden dann im fünften Abschnitt vorgestellt, in dem auch die Implementierung beschrieben wird. Die Implementierung ist jedoch auf Unitklauseln eingeschränkt, so dass dort auch noch der Superpositionskalkül für Unitklauseln eingeführt wird. Der siebte Abschnitt enthält schließlich eine Auswertung der mit der Implementierung erlangten Messergebnisse, die erste Schlüsse über den Nutzen von Constraints zulassen. Abgeschlossen wird die Arbeit durch eine Zusammenfassung und einen Ausblick auf Möglichkeiten für weitere Arbeiten.

1. Grundlegende Definitionen

Die im Folgenden vorgestellten grundlegenden Definitionen und Schreibweisen sind im Wesentlichen analog zu [Ave95] und [GN01]. Dort findet sich auch eine detailliertere Darstellung der meisten vorgestellten Konzepte.

1.1. Terme, Gleichungen, Klauseln, Closures, Substitutionen

In diesem Abschnitt sollen zunächst die syntaktischen Konstrukte, die im Folgenden benutzt werden, definiert werden.

Sei $\mathcal{F}$ eine Menge von *Funktionssymbolen*, $\mathcal{X}$ eine unendliche Menge von *Variablen* und α eine totale Funktion $\alpha\colon \mathcal{F} \to \mathbb{N}$, die jedem Funktionssymbol eine *Stelligkeit* zuordnet. Ein Funktionssymbol f mit $\alpha(f) = n$ wird *n-stelliges Funktionssymbol* genannt. 0-stellige Funktionssymbole heißen *Konstanten*. Ein *Term* über $\mathcal{F}$ und $\mathcal{X}$ ist entweder eine Variable $x \in \mathcal{X}$ oder ein Ausdruck der Form $f(t_1, \ldots t_n)$, wobei $f \in \mathcal{F}$ ein n-stelliges Funktionssymbol und $t_1, \ldots, t_n$ Terme über $\mathcal{F}$ und $\mathcal{X}$ sind. Die Menge aller Terme über $\mathcal{F}$ und $\mathcal{X}$ wird mit $T(\mathcal{F}, \mathcal{X})$ bezeichnet. Zwei Terme s und t heißen *syntaktisch äquivalent* (Schreibweise: $s \equiv t$), falls sie das gleiche Variablensymbol sind oder sie die Form $f(s_1, \ldots, s_n)$ und $g(t_1, \ldots, t_n)$ besitzen, f und g das gleiche Funktionssymbol sind und $s_1 \equiv t_1, \ldots, s_n \equiv t_n$ gilt.

Für die weiteren Definitionen wird das Konzept einer *Multimenge* benötigt. Eine *Multimenge M über einer Menge Z* ist eine Funktion $M : Z \to \mathbb{N}$, wobei nur für endlich viele $z \in Z$ gilt $M(z) > 0$. Multimengen werden häufig ähnlich wie Mengen geschrieben, mit dem Unterschied, dass Elemente mehrfach vorkommen dürfen. Beispiel: Die Multimenge M über $\mathbb{N}$, die über $M(1) = 2$, $M(2) = 1$, $M(3) = 4$ und $M(i) = 0$ sonst gegeben ist, lässt sich auch Schreiben als $\{1, 1, 2, 3, 3, 3, 3\}$. Für zwei Multimengen M_1 und M_2 über einer Menge Z gilt $M_1 = M_2$, wenn für alle $z \in Z$ gilt $M_1(z) = M_2(z)$. Die Vereinigung $M := M_1 \cup M_2$ zweier Multimengen M_1, M_2 über einer Menge Z ist definiert über $M(x) = M_1(x) + M_2(x)$ für alle $x \in Z$.

Eine *Gleichung* ist ein Ausdruck der Form $s \simeq t$, wobei s und t Terme sind. Zwei Gleichungen $s \simeq t$ und $u \simeq v$ sind genau dann syntaktisch äquivalent, wenn $s \equiv u$ und $t \equiv v$ gilt oder wenn $s \equiv v$ und $t \equiv u$ gilt. Eine *negierte Gleichung* oder *Ungleichung* ist ein Ausdruck der Form $s \not\simeq t$. Zwei Ungleichungen $s \not\simeq t$ und $u \not\simeq v$ sind genau dann syntaktisch äquivalent, wenn $s \equiv u$ und $t \equiv v$ gilt oder wenn $s \equiv v$ und $t \equiv u$ gilt.

Ein *Literal* der reinen Gleichheitslogik erster Stufe ist eine Gleichung oder Ungleichung. Falls es unwichtig ist, ob es sich um eine Gleichung oder Ungleichung handelt, wird im Folgenden für Literale die Schreibweise $s \dot{\simeq} t$ verwendet, wobei s und t Terme sind. Wegen dieser Definition werden Gleichungen auch *positive Literale* und Ungleichungen *negative Literale* genannt. Die Funktion *lmul*, die Literale auf Multimengen von Termen abbildet, sei definiert durch $\mathit{lmul}(s \simeq t) := \{s, t\}$ und $\mathit{lmul}(s \not\simeq t) := \{s, s, t, t\}$. Zwei Literale l_1 und l_2 (Schreibweise $l_1 \equiv l_2$) heißen genau dann syntaktisch äquivalent, wenn $lmul(l_1) = lmul(l_2)$ gilt. Diese Definition der syntaktischen Äquivalenz auf Literalen entspricht der obigen Definition der syntaktischen Äquivalenz auf Gleichungen und Ungleichungen. Die Funktion *lmul* ermöglicht eine einfache Definition der syntaktischen Äquivalenz, wird jedoch auch noch im Folgenden benötigt werden.

Eine *Klausel* ist eine Multimenge von Literalen. Üblicherweise werden Klauseln ohne Mengenklammern geschrieben. Sie lassen sich auch in der Form $\Gamma \to \Delta$ schreiben, wobei Γ die Multimenge der Ungleichungen (die jedoch dann als Gleichungen geschrieben werden) und Δ die Multimenge der Gleichungen ist. Beispielsweise lässt sich die Klausel $\{s_1 \not\simeq t_1, s_1 \not\simeq t_1, s_2 \simeq t_2, s_3 \not\simeq t_3, s_4 \not\simeq t_4\}$ auch in der Form $s_1 \not\simeq t_1, s_1 \not\simeq t_1, s_2 \simeq t_2, s_3 \not\simeq t_3, s_4 \not\simeq t_4$ oder $s_1 \simeq t_1, s_1 \simeq t_1, s_3 \simeq t_3, s_4 \simeq t_4 \to s_2 \simeq t_2$ schreiben. Bei einer Klausel $\Gamma \to \Delta$ wird Γ *Antezedent* und Δ *Sukzedent* der Klausel genannt. Die Klausel, in der Γ und Δ leer sind, wird die *leere Klausel* $\Box$ genannt. Eine *Hornklausel* ist eine Klausel, bei der Δ höchstens eine Gleichung enthält. Eine Hornklausel ist also eine Klausel, die maximal ein positives Literal enthält. Die Funktion *cmul*, die eine Klausel $\{l_1, \ldots, l_n\}$ auf eine Multimenge von Multimengen von Termen abbildet, ist definiert durch $cmul(\{l_1, \ldots, l_n\}) := \{lmul(l_1), \ldots, lmul(l_n)\}$. Die *syntaktische Äquivalenz zweier Klauseln* C und D wird als $C \equiv D$ geschrieben und ist wie folgt definiert: $C \equiv D$ gdw. $cmul(C) = cmul(D)$. Zum Beispiel sind die Klauseln $s_1 \simeq t_1, s_2 \not\simeq t_2$ und $t_2 \not\simeq s_2, s_1 \simeq t_1$ syntaktisch äquivalent, denn es gilt $cmul(s_1 \simeq t_1, s_2 \not\simeq t_2) = \{lmul(s_1 \simeq t_1), lmul(s_2 \not\simeq t_2)\} = \{\{s_1, t_1\}, \{s_2, s_2, t_2, t_2\}\} = cmul(t_2 \not\simeq s_2, s_1 \simeq t_1)$.

Eine *Position* ist entweder λ oder besitzt die Form $n.p$, wobei n eine natürliche Zahl größer 0 und p eine Position ist. Für einen Term t und eine Position p sei $t|_p$ definiert durch $t|_p := t$, falls $p = \lambda$, und $t|_p := t_j|_{p'}$, falls $t \equiv f(t_1, \ldots, t_j, \ldots, t_n)$ und $p = j.p'$ mit $1 \leq j \leq n$. Ansonsten ist $t|_p$ undefiniert. Für eine Position p und einen Term t nennt man $t|_p$ den *Teilterm von t an der Position p*. Gilt $p \neq \lambda$, so heißt $t|_p$ *echter Teilterm* von t. Diese Definition wird durch $s \doteq t|_{1.p} := s|_p$ und $s \doteq t|_{2.p} := t|_p$ auf Literale und durch $(l_1 \vee \ldots \vee l_n)|_{j.p} := l_j|_p$ für $1 \leq j \leq n$ auf Klauseln fortgesetzt. Beachte: Für zwei Klauseln C und D mit $C \equiv D$ und eine Position p gilt im Allgemeinen nicht $C|_p \equiv D|_p$!

Sei $t[s]_p$ der Term, den man durch Ersetzen des Teilterms von t an der Position p durch den Term s erhält. Diese Definition gelte analog für Literale und Klauseln. Eine Variable x *kommt* an einer Position p eines Terms t *vor*, falls $t|_p \equiv x$ gilt; p heißt dann *Variablenposition*. Die Menge aller Variablen, die in t vorkommen, wird mit $var(t)$ bezeichnet. Diese Definition wird auf natürliche Art auf Literale und Klauseln erweitert. Zwei Terme (Literale, Klauseln) s und t heißen genau dann *variablendisjunkt*, wenn $var(s) \cap var(t) = \emptyset$ gilt. Beispiel: Für $t \equiv f(x, g(b, h(y)), d)$ ist $t|_{2.2} \equiv h(y)$ und $t[c]_{2.2} \equiv f(x, g(b, c), d)$. Die Variable y kommt in t an der Position 2.2.1 vor und $var(t) = \{x, y\}$.

Eine *Substitution* über einer Menge von Funktionssymbolen $\mathcal{F}$ und einer Variablenmenge $\mathcal{X}$ ist eine totale Funktion $\sigma : \mathcal{X} \to T(\mathcal{F}, \mathcal{X})$, wobei $x\sigma = x$ für fast alle $x \in \mathcal{X}$ gilt (Schreibweise: Die Substitution σ, die gegeben ist durch $x_i\sigma = t_i$ für alle $1 \leq i \leq n$ und $x\sigma = x$ für alle anderen Variablen x, wird in der Form $\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$ geschrieben). Gilt $x\sigma \neq x$ für eine Variable $x \in \mathcal{X}$, so heißt x *gebunden* in σ. Die Menge aller in σ gebundenen Variablen wird mit $dom(\sigma)$ bezeichnet. Die Substitution, die keine gebundenen Variablen besitzt, die also jede Variable auf sich selbst abbildet, wird mit *id* bezeichnet. Zwei Substitutionen σ_1, σ_2 heißen *syntaktisch äquivalent* (Schreibweise: $\sigma_1 \equiv \sigma_2$) genau dann, wenn für alle Variablen $x \in \mathcal{X}$ gilt $x\sigma_1 \equiv x\sigma_2$. Die Substitution σ wird durch $f(t_1, \ldots, t_n)\sigma := f(t_1\sigma, \ldots, t_n\sigma)$ auf Terme, durch $(s \mathrel{\dot{\simeq}} t)\sigma := s\sigma \mathrel{\dot{\simeq}} t\sigma$ auf Literale und schließlich durch $(l_1, \ldots, l_n)\sigma := l_1\sigma, \ldots, l_n\sigma$ auf Klauseln fortgesetzt. Das Ergebnis der Anwendung einer Substitution σ auf eine Klausel (ein Literal, einen Term) heißt *Instanz* der Klausel (des Literals, des Terms). Instanzen, die keine Variablen enthalten, heißen *Grundinstanzen*, die zugehörigen Substitutionen *Grundsubstitutionen* bezüglich der Klausel (des Literals, des Terms). Terme und Klauseln, die keine Variablen enthalten, heißen entsprechend *Grundterme* und *Grundklauseln*. Eine Grundklausel enthält nur Grundterme. Die Menge aller Grundterme über $\mathcal{F}$ wird mit $T(\mathcal{F})$ bezeichnet.

Beispiel Sei $C \equiv f(x) \simeq b$ eine Klausel und seien $\sigma_1 = \{x \leftarrow f(y)\}$ sowie $\sigma_2 = \{x \leftarrow a\}$ zwei Substitutionen. Dann sind $C\sigma_1 \equiv f(f(y)) \simeq b$ und $C\sigma_2 \equiv f(a) \simeq b$ zwei Instanzen von C. Dabei ist $C\sigma_2$ im Gegensatz zu $C\sigma_1$ eine Grundinstanz von C und entsprechend ist σ_2 bezüglich C eine Grundsubstitution.

Eine Substitution σ heißt *Variablenumbenennung*, wenn σ injektiv ist und für alle Variablen $x \in \mathcal{X}$ gilt, dass $x\sigma$ eine Variable ist. Jede Variablenumbenennung ist bijektiv, da sie nach Definition injektiv ist und da es für jede Substitution σ nur endliche viele Variablen x mit $x\sigma \neq x$ gibt. Zwei Klauseln C und D heißen *syntaktisch äquivalent bis auf Variablenumbenennung* genau dann, wenn es eine Variablenumbenennung σ gibt, so dass $C\sigma \equiv D$ gilt. Da solche Klauseln im Folgenden gleich behandelt werden, wird für eine Klausel C und eine Klauselmenge M auch $C \in M$ geschrieben, wenn eine zu C bis auf Variablenumbenennung syntaktisch äquivalente Klausel im üblichen Sinne in M enthalten ist.

Im Folgenden wird es häufig nötig sein, nach der Anwendung einer Substitution σ auf eine Klausel C weiterhin die Variablenpositionen von C zu kennen. Dazu wird das Konzept der *Closure* eingeführt. Unter einer *Closure* $C{\cdot}\sigma$ versteht man für eine Klausel C und eine Substitution σ die Instanz $C\sigma$ von C, die aber zusätzlich die Information enthält, aus welcher Klausel C und Substitution σ sie entstanden ist. Unter einer *Substitutionsposition* p einer Closure $C{\cdot}\sigma$ versteht man eine Position von $C\sigma$, für die $C\sigma|_p$ keine Variable ist und es einen Präfix p' von p gibt, für den p' Variablenposition von C ist. Mit $\mathit{SubPos}(C{\cdot}\sigma)$ wird die Menge aller Substitutionspositionen von $C{\cdot}\sigma$ bezeichnet. Da es teilweise nötig ist, auch über die Substitutionspositionen von Teilen einer Closure zu sprechen, werden die Schreibweisen $\cdot$ und *SubPos* auch für Literale und Terme verwendet. Dies ist dann aber immer im Kontext einer Closure zu sehen. Da es sich bei Closures um Instanzen von Klauseln mit zusätzlichen Informationen handelt, werden sie im Folgenden oft auch einfach als Instanzen behandelt und bezeichnet. Die syntaktische Äquivalenz wird jedoch abweichend wie folgt definiert: Zwei Closures $C{\cdot}\sigma_1$ und $D{\cdot}\sigma_2$ sind genau dann *syntaktisch äquivalent*, wenn $C\sigma_1 \equiv D\sigma_2$ und $\mathit{SubPos}(C{\cdot}\sigma_1) = \mathit{SubPos}(D{\cdot}\sigma_2)$ gilt.

Beispiel Sei $C := f(x, y) \simeq b$ eine Klausel und $\sigma = \{x \leftarrow a, y \leftarrow g(g(z))\}$ eine Substitution. Die Closure $C{\cdot}\sigma$ entspricht dann der Klausel $f(a, g(g(z))) \simeq b$, wobei zusätzlich die Information, dass an den Positionen 1.1.1 und 1.1.2 eine Substitution stattgefunden hat, enthalten ist. Die Positionen 1.1.1 und 1.1.2 sind also

Substitutionspositionen. Ebenso ist die Position 1.1.2.1 eine Substitutionsposition. Die Position 1.1.2.1.1 ist jedoch keine Substitutionsposition, da es sich um eine Variablenposition handelt. Markiert man die Substitutionspositionen von $C{\cdot}\sigma$ durch Unterstreichung, so kann man sich also $C{\cdot}\sigma$ auch in der folgenden Form vorstellen: $f(\underline{a}, \underline{g(g}(z))) \simeq b$. Man beachte: In dieser Schreibweise gilt $f(\underline{a}, \underline{g(g}(z))) \simeq b \not\equiv f(a, g(g(z))) \simeq b$. Für den Teilterm $f(x, y){\cdot}\sigma$ von $C{\cdot}\sigma$, also für $f(\underline{a}, \underline{g(g}(z)))$ gilt $SubPos(f(x, y){\cdot}\sigma) = \{1.1, 1.2, 1.2.1\}$.

Die Komposition σ zweier Substitutionen σ_1, σ_2 (Schreibweise: $\sigma := \sigma_1 \circ \sigma_2$) ist definiert als $x\sigma := x\sigma_2\sigma_1$ für alle Variablen x. Lässt sich σ so darstellen, so heißt σ *Instanz* von σ_2. Gilt $\sigma = \sigma \circ \sigma$, so heißt die Substitution σ *idempotent*. Eine Substitution σ mit $s\sigma \equiv t\sigma$ für zwei Terme s, t heißt *Unifikator von s und t*; man sagt dann σ *unifiziert* s und t. Ein Unifikator einer Menge von Termpaaren $M = \{(s_1, t_1), \ldots, (s_n, t_n)\}$ ist eine Substitution, die jedes Termpaar s_i, t_i der Menge M unifiziert. Ein *allgemeinster Unifikator* zweier Terme s und t bzw. einer Menge von Termpaaren M ist ein Unifikator σ von s und t bzw. M mit der Eigenschaft, dass jeder andere Unifikator von s und t bzw. M eine Instanz von σ ist (Schreibweise: $mgu(s, t) = \sigma$ bzw. $mgu(M) = \sigma$). Der allgemeinste Unifikator ist bis auf Variablenumbenennungen eindeutig bestimmt. Existiert ein Unifikator zweier Terme s und t bzw. einer Menge von Termpaaren M, so heißen diese Terme bzw. diese Menge *unifizierbar*.

Beispiel Die beiden Terme $t_1 = f(g(a, x))$ und $t_2 = f(y)$ sind unifizierbar, denn z. B. für die Substitution $\sigma = \{x \leftarrow b, y \leftarrow g(a, b)\}$ gilt $t_1\sigma \equiv t_2\sigma$. Dieser Unifikator σ ist jedoch kein allgemeinster Unifikator, denn σ ist Instanz des Unifikators $\theta = \{y \leftarrow g(a, x)\}$. Der Unifikator θ ist dagegen ein allgemeinster Unifikator von t_1 und t_2. Die bis auf Variablenumbenennung zu θ syntaktisch äquivalente Substitution $\{x \leftarrow z, y \leftarrow g(a, z)\}$ ist ebenfalls ein allgemeinster Unifikator von t_1 und t_2.

1.2. Semantik

Nachdem nun die syntaktischen Elemente der reinen Gleichheitslogik erster Stufe definiert wurden, wird im Folgenden deren Semantik festgelegt.

Eine *Kongruenz* $\approx$ auf einer Menge D und einer Menge von Funktionen F ist eine Teilmenge von D^2 mit folgenden Eigenschaften:

- $x \approx x$ für alle $x \in D$ (Reflexivität)
- wenn $x \approx y$ gilt, dann gilt auch $y \approx x$ für alle $x, y \in D$ (Symmetrie)
- wenn $x \approx y$ und $y \approx z$ gelten, dann gilt auch $x \approx z$ für alle $x, y, z \in D$ (Transitivität)
- wenn $x_1 \approx y_1$ und $\ldots$ und $x_n \approx y_n$ gelten, dann gilt auch $f(x_1, \ldots x_n) \approx f(y_1, \ldots, y_n)$ für alle $x_1, \ldots, x_n, y_1, \ldots, y_n \in D$ und alle $f : D^n \to D \in F$ (Monotonie)

Eine *Interpretation* $\mathcal{J}$ in der reinen Gleichheitslogik erster Stufe für eine Menge von Funktionssymbolen $\mathcal{F}$ besteht aus drei Teilen:

- einer nicht leeren Trägermenge $D_{\mathcal{J}}$
- einer Funktion $f_{\mathcal{J}} : D_{\mathcal{J}}^n \to D_{\mathcal{J}}$ für jedes n-stellige Funktionssymbol $f \in \mathcal{F}$
- einer Kongruenz $\approx_{\mathcal{J}}$ auf $D_{\mathcal{J}}$

Eine *Variablenbelegung* (kurz *Belegung*) für eine Variablenmenge $\mathcal{X}$ und eine Interpretation $\mathcal{J}$ ist eine totale Funktion $bel : \mathcal{X} \to D_{\mathcal{J}}$, also eine Funktion, die jeder Variable ein Element der Trägermenge zuweist.

Eine Interpretation $\mathcal{J}$ *erfüllt eine Gleichung* $s \simeq t$ (Schreibweise: $\mathcal{J} \models s \simeq t$) genau dann, wenn für alle Belegungen $eval(s) \approx_{\mathcal{J}} eval(t)$ gilt, wobei die Funktion $eval : T(\mathcal{F}, \mathcal{X}) \to D_{\mathcal{J}}$ gegeben ist durch:

- $eval(x) = bel(x)$ falls $x \in \mathcal{X}$
- $eval(f(t_1, \ldots, t_n)) = f_{\mathcal{J}}(eval(t_1), \ldots, eval(t_n))$ sonst

Eine Interpretation $\mathcal{J}$ *erfüllt eine Ungleichung* $s \not\simeq t$ (Schreibweise: $\mathcal{J} \models s \not\simeq t$) genau dann, wenn nicht $\mathcal{J} \models s \simeq t$ gilt (Schreibweise: $\mathcal{J} \not\models s \simeq t$). *Eine Klausel* C wird genau dann von $\mathcal{J}$ *erfüllt* (Schreibweise: $\mathcal{J} \models C$), wenn mindestens eine Gleichung aus dem Antezedenten nicht von $\mathcal{J}$ erfüllt wird oder mindestens eine Gleichung des Sukzedenten erfüllt wird. Klauseln können also als Disjunktion von Literalen betrachtet werden. Existiert für eine Klausel C eine Interpretation $\mathcal{J}$ mit $\mathcal{J} \models C$, so heißt die Klausel *erfüllbar*, ansonsten *unerfüllbar*. Eine Klausel C heißt *Tautologie*, wenn für alle Interpretationen $\mathcal{J}$ gilt: $\mathcal{J} \models C$. Die Interpretation $\mathcal{J}$ ist ein *Modell* für eine Menge von Klauseln S (Schreibweise: $\mathcal{J} \models S$) genau dann, wenn $\mathcal{J} \models C$ für alle Klauseln $C \in S$ gilt. Existiert für eine Klauselmenge S eine Interpretation $\mathcal{J}$ für die $\mathcal{J} \models S$ gilt, so heißt S *erfüllbar*, anderenfalls *unerfüllbar* oder *widersprüchlich*. Die leere Klausel $\square$ ist unerfüllbar. Das heißt, dass jede Menge, die die leere Klausel enthält, widersprüchlich ist.

Eine Klausel C *folgt aus einer Klauselmenge* M genau dann, wenn jede Interpretation $\mathcal{J}$, die alle Klauseln aus M erfüllt, auch C erfüllt. Wie angekündigt soll sich dieser Text mit automatischem Theorembeweisen beschäftigen. Die grundlegende Fragestellung ist also, ob aus einer gegebenen Menge von Klauseln, den sogenannten *Axiomen*, eine andere Klausel, *Konklusion* genannt, in dieser Art folgt.

1.3. Erfüllbarkeitsproblem und Herbrand Interpretationen

Das *Erfüllbarkeitsproblem* ist die Frage, ob eine Klauselmenge ein Modell besitzt, also erfüllbar ist. Das hier betrachtete Problem, ob aus einer Menge von Klauseln S' eine bestimmte Klausel $C \equiv l_1, \ldots, l_n$ folgt, ist äquivalent zu der Frage, ob die Klauselmenge $S := S' \cup \{\{\neg l_1\}, \ldots, \{\neg l_n\}\}$ widersprüchlich ist. Das Erfüllbarkeitsproblem ist für die reine Gleichheitslogik erster Stufe nicht entscheidbar. Es ist aber rekursiv aufzählbar, ob eine Klauselmenge widersprüchlich ist. Damit ist auch die Fragestellung nicht entscheidbar, aber rekursiv aufzählbar.

Wie kann man vorgehen, wenn man wissen will, ob eine gegebene Klauselmenge S erfüllbar ist? Es ist nicht möglich, nacheinander alle möglichen Interpretationen zu testen, da es unendlich viele Interpretation gibt. Es genügt jedoch, eine wesentlich kleinere, aber leider trotzdem unendliche Menge von Interpretationen zu testen, da S genau dann erfüllbar ist, wenn es eine sogenannte *Herbrand Interpretation* gibt, die S erfüllt. Eine *Herbrand Interpretation* $\mathcal{J}$ ist eine Interpretation mit $D_{\mathcal{J}} = T(\mathcal{F})$ und $f_{\mathcal{J}}(t_1, \dots t_n) = f(t_1, \dots t_n)$ für alle $f \in \mathcal{F}$ (das heißt die Funktionssymbole werden durch ihre syntaktische Darstellung interpretiert). Eine Herbrand Interpretation in der reinen Gleichheitslogik erster Stufe wird also nur noch durch eine Kongruenz auf den Grundtermen $T(\mathcal{F})$ bestimmt.

Eine Möglichkeit, eine Herbrand Interpretation für eine gegebene Menge von Funktionssymbolen $\mathcal{F}$ anzugeben, besteht darin, eine Relation $\mathcal{R}$ auf den Termen $T(\mathcal{F}, \mathcal{X})$ in Form einer Teilmenge von $T(\mathcal{F}, \mathcal{X})^2$ anzugeben. Die für die Herbrand Interpretation nötige Kongruenz $\mathcal{R}^* \subseteq T(\mathcal{F})^2$ sei die kleinste Kongruenz mit $(s\sigma, t\sigma) \in \mathcal{R}^*$ für alle $(s, t) \in \mathcal{R}$ und alle Grundsubstitutionen σ. Da sie eine eindeutige Herbrand Interpretation implizieren, werden im Folgenden $\mathcal{R}$ und $\mathcal{R}^*$ je nach Kontext als Herbrand Interpretation behandelt.

1.4. Reduktionssysteme

In der Praxis stellt sich das Problem, zu bestimmen, ob in der von $\mathcal{R}$ implizierten Kongruenz $\mathcal{R}^*$ zwei Grundterme s und t kongruent sind, also ob $(s, t) \in \mathcal{R}^*$ gilt. Wie man leicht sieht, ist das sogenannte *Wortproblem*, also das Problem zu bestimmen, ob zwei Elemente einer Menge in einer Äquivalenzrelation äquivalent sind, auf dieses Problem reduzierbar. Da das Wortproblem unentscheidbar ist, ist auch das Problem unentscheidbar, ob in der von $\mathcal{R}$ implizierten Kongruenz $\mathcal{R}^*$ zwei Grundterme kongruent sind.

Eine mögliche Lösung dieses Problems ist es, $\mathcal{R}$ als sogenanntes Reduktionssystem R aufzufassen (Schreibweise: wird $\mathcal{R}$ als Reduktionssystem und nicht als Herbrand Interpretation betrachtet, wird R statt $\mathcal{R}$ verwendet). Besitzt dieses Reduktionssystem R eine zusätzliche Eigenschaft, die sogenannte Grundkonvergenz, so ist es entscheidbar, ob zwei Grundterme kongruent sind.

Vor der genauen Definition, was ein Reduktionssystem ist, werden zunächst noch einige grundlegende Definitionen für Relationen benötigt. Für eine Relation $\Rightarrow$ sei die Relation $\Leftarrow$ gegeben durch $s \Leftarrow t$ genau dann, wenn $t \Rightarrow s$; weiter sei die Relation $\Leftrightarrow$ gegeben durch $s \Leftrightarrow t$ genau dann, wenn $s \Rightarrow t$ oder $s \Leftarrow t$. Für eine Relation $\Rightarrow$ bezeichne $\Rightarrow^*$ ihren reflexiv-transitiven Abschluss. Falls $s \Rightarrow^* t$ und kein t' mit $t \Rightarrow t'$ existiert, dann heißt t *Normalform von s bezüglich* $\Rightarrow$ (Schreibweise: $t \equiv s\Downarrow$). Die Relation $\Rightarrow$ heißt *terminierend*, falls es keine unendliche Folge $s_1 \Rightarrow s_2 \Rightarrow \dots$ gibt. Die Relation heißt *konfluent*, falls aus $s_1 \Leftarrow^* t \Rightarrow^* s_2$ folgt, dass ein u existiert mit $s_1 \Rightarrow^* u \Leftarrow^* s_2$. Folgt aus $s_1 \Leftarrow t \Rightarrow s_2$, dass ein u existiert mit $s_1 \Rightarrow^* u \Leftarrow^* s_2$, so heißt sie *lokal konfluent*. Terminierende, lokal konfluente Relationen sind konfluent. Eine terminierende, konfluente Relation heißt *konvergent*.

Eine *Regel* ist ein gerichtetes Paar von Termen (l, r) (Schreibweise: $l \Rightarrow r$). Ein *Reduktionssystem* R ist eine Menge von Regeln. Die durch R gegebene *Reduktionsrelation* $\Rightarrow_R$ auf $T(\mathcal{F}, \mathcal{X})$ ist die kleinste mit der Termstruktur verträgliche Relation mit $l\sigma \Rightarrow_R r\sigma$ für alle $l \Rightarrow r \in R$ und alle Substitutionen σ. Eine Relation $\Rightarrow$ heißt hierbei *verträglich mit der Termstruktur*, falls für beliebige Terme l, r, t und für alle Positionen p von t gilt, dass aus $l \Rightarrow r$ auch $t[l]_p \Rightarrow t[r]_p$ folgt.

Für die von einer Relation $\mathcal{R}$ implizierte Kongruenz $\mathcal{R}^*$ gilt mit diesen Definitionen $(l, r) \in \mathcal{R}^*$ genau dann, wenn $l \Leftrightarrow_R r$. Also ist diese Definitionen in Hinblick auf das ursprüngliches Problem sinnvoll. Es stellt sich also das Problem zu bestimmen, wann $l \Leftrightarrow_R r$ gilt.

Ein Term t heißt genau dann *reduzierbar an einer Position* p durch eine Regel $l \Rightarrow r$, wenn es eine Substitution σ gibt mit $t|_p \equiv l\sigma$. Ein Term t ist somit an einer Position p durch eine Regel $l \Rightarrow r$ mit einer Substitution σ reduzierbar genau dann, wenn für jedes Reduktionssystem R mit $l \Rightarrow r \in R$ die Beziehung $t \Rightarrow_R t[r\sigma]_p$ gilt. Ein Term t heißt genau dann *reduzierbar durch eine Regel*, wenn es eine Position p von t gibt, an der t durch die Regel reduzierbar ist. Ein Term heißt *reduzierbar bezüglich eines Reduktionssystems*, wenn es eine Regel in dem Reduktionssystem gibt, durch die der Term reduzierbar ist. Diese Definitionen werden auf Literale und Klauseln erweitert, indem nur Positionen betrachtet werden, die auf einen Term verweisen.

Beispiel Der Term $s(f(a, b))$ ist an der Position 1 reduzierbar durch die Regel $f(x, b) \Rightarrow x$. Denn die Substitution $\sigma = \{x \leftarrow a\}$ erfüllt die Bedingung $s(f(a, b))|_1 \equiv f(a, b) \equiv f(x, b)\sigma$. Für $R = \{f(x, b) \Rightarrow x\}$ gilt somit $s(f(a, b)) \Rightarrow_R s(a)$.

R heißt terminierend, konfluent, konvergent oder lokal konfluent, falls $\Rightarrow_R$ die entsprechende Eigenschaft besitzt. Bezüglich eines konvergenten Reduktionssystems R besitzt jeder Term t eine eindeutige Normalform $t{\Downarrow_R}$ und es gilt $s \Leftrightarrow_R^* t$ genau dann, wenn $s{\Downarrow_R} \equiv t{\Downarrow_R}$. Für konvergente Reduktionssysteme mit entscheidbarer Reduktionsrelation $\Rightarrow_R$ ist somit entscheidbar, ob $s \Leftrightarrow_R^* t$ gilt. Für die folgenden Beweise genügt es jedoch meist, wenn das Reduktionssystem *grundkonvergent* ist, das heißt die Eigenschaft der Konvergenz lediglich für Grundterme gefordert wird.

Wenn man $\mathcal{R}$ als Reduktionssystem auffasst, ist – falls R grundkonvergent ist – das Problem gelöst, zu bestimmen, ob in der von $\mathcal{R}$ implizierten Kongruenz zwei Grundterme s und t äquivalent sind. Es entsteht jedoch das neue Problem, die Konvergenz, also die Termination und Konfluenz, von R nachzuweisen.

1.5. Konvergenz von Reduktionssystemen und Reduktionsordnungen

Zunächst soll die Termination von Reduktionssystemen betrachtet werden. Die Frage, ob ein Reduktionssystem terminierend ist, ist unentscheidbar. Trotzdem gibt es mit den sogenannten *Reduktionsordnungen* eine Methode, die Termination eines Reduktionssystems zu garantieren.

Eine Partialordnung $>$ auf einer Menge M ist eine Relation auf M^2 mit folgenden Eigenschaften:

- $x \not> x$ für alle $x \in M$ (Irreflexivität)
- gilt $x > y$, so gilt auch $y \not> x$ für alle $x, y \in M$ (Antisymmetrie)
- gilt $x > y$ und $y > z$, so gilt auch $x > z$ für alle $x, y, z \in M$ (Transitivität)

Beispielsweise ist die Relation $>$ auf natürlichen Zahlen eine Partialordnung. Aber auch z. B. die folgende Relation $>_p$ auf Positionen ist eine Partialordnung: Es gilt

$p >_p \lambda$ für alle Positionen $p \neq \lambda$ und für zwei Positionen $p_1 = i_1.r_1$ und $p_2 = i_2.r_2$ mit $i_1, i_2 \in \mathbb{N}$ gilt genau dann $p_1 >_p p_2$, wenn $i_1 > i_2$ oder wenn $i_1 = i_2$ und $r_1 >_p r_2$ gilt.

Eine *Reduktionsordnung* $\succ$ ist eine Partialordnung auf Termen, die zusätzlich die folgenden Eigenschaften besitzt:

- es gibt keine unendliche Folge $s_1 \succ s_2 \succ \ldots$ (Termination)
- gilt $s \succ t$, so gilt auch $s\sigma \succ t\sigma$ für alle $s, t \in T(\mathcal{F}, \mathcal{X})$ und alle Substitutionen σ (Verträglichkeit mit Substitutionen)
- gilt $t_1 \succ t_2$, so gilt auch $t[t_1]_p \succ t[t_2]_p$ für alle $t, t_1, t_2 \in T(\mathcal{F}, \mathcal{X})$ und alle Positionen p von t (Verträglichkeit mit der Termstruktur)

Ein Reduktionssystem R ist genau dann terminierend, wenn es eine Reduktionsordnung $\succ$ gibt, für die $R \subseteq \succ$, das heißt $s \succ t$ für alle Regeln $s \Rightarrow t \in R$, gilt. Wenn man also zeigen kann, dass es eine solche Reduktionsordnung gibt, braucht man nur noch die lokale Konfluenz des Reduktionssystems nachzuweisen, um die Konvergenz des Reduktionssystems zu zeigen, da jedes terminierende, lokal konfluente Reduktionssystem konfluent ist. Auf diese Art wird im Folgenden die Konvergenz von Reduktionssystemen gezeigt werden. Reduktionsordnungen werden jedoch auch für andere Zwecke eingesetzt werden, für die weitere Eigenschaften notwendig sind. Daher werden hier nur grundtotale Reduktionsordnungen verwendet, die die *Teiltermeigenschaft* besitzen, das heißt für die $t \succ s$ für alle echten Teilterme s eines Terms t gilt.

Zu einer Partialordnung $>$ auf einer Menge M ist die Relation $\geq$ gegeben durch $\geq := > \cup \{(t,t) \mid t \in M\}$. Eine Partialordnung, bei der für alle $s, t \in M$ mit $s \neq t$ entweder $s > t$ oder $t > s$ gilt, heißt *total*. Besitzt eine Partialordnung auf Termen (Literalen, Klauseln) diese Eigenschaft für Grundterme (-literale, -klauseln), so heißt sie *grundtotal*.

Weiterhin ist zu einer Partialordnung $>$ auf einer Menge M ihre *Multimengenerweiterung* $>_{mul}$ als die kleinste Partialordnung auf Multimengen über M definiert, die $Z \cup \{s\} >_{mul} Z \cup \{t_1, \ldots, t_n\}$ für jede Multimenge Z über M und $s, t_1, \ldots, t_n \in M$ erfüllt, falls $s > t_i$ für alle $1 \leq i \leq n$ mit $n \geq 0$. Ist die Partialordnung $>$ total, grundtotal oder terminierend, so ist es auch ihre Multimengenerweiterung $>_{mul}$.

Dies wird benutzt, um eine Reduktionsordnung $\succ$ von Termen auf Literale und Klauseln zu erweitern. Für die Reduktionsordnung $\succ$ sei ihre Erweiterung auf Literale $\succ_l$ definiert durch $l_1 \succ_l l_2$ genau dann, wenn $lmul(l_1) \succ_{mul} lmul(l_2)$. Ähnlich erfolgt die Erweiterung auf Klauseln. Die Erweiterung der Reduktionsordnung $\succ$ auf Klauseln $\succ_c$ ist gegeben über $C_1 \succ_c C_2$ genau dann, wenn $cmul(C_1)\ (\succ_{mul})_{mul}\ cmul(C_2)$. Die Verfeinerung $\succ_{cl}$ dieser Ordnung für Closures wird wie folgt definiert: Für zwei Closures $C_1 \cdot \sigma$ und $C_2 \cdot \sigma$ gilt genau dann $C_1 \cdot \sigma \succ_{cl} C_2 \cdot \sigma$, wenn $C_1\sigma \succ_c C_2\sigma$ gilt oder wenn $C_1\sigma \equiv C_2\sigma$ und $SubPos(C_2 \cdot \sigma)\ (>_p)_{mul}\ SubPos(C_1 \cdot \sigma)$. Die Ordnungen $\succ_l$ und $\succ_c$ sind grundtotal und terminierend. Auch $\succ_{cl}$ ist terminierend, da diese Ordnung eine lexikographische Kombination aus den beiden terminierenden Ordnungen $\succ_c$ und $(>_p)_{mul}$ ist. Außerdem ist $\succ_{cl}$ grundtotal, da $\succ_c$ grundtotal und $(>_p)_{mul}$ total ist.

Ein Term t heißt *maximal* bezüglich einer Klausel beziehungsweise eines Literals, wenn für jeden anderen Term s der Klausel beziehungsweise des Literals gilt $s \not\succ t$.

Entsprechend heißt ein Literal l *maximal* bezüglich einer Klausel, wenn für jedes andere Literal l' der Klausel $l' \not\succ l$ gilt. Gilt sogar $l \succ l'$ bzw. $t \succ s$, so heißt das Literal bzw. der Term *strikt maximal*.

Abschließend noch einige Schreibweisen: Für eine Partialordnung $>$ steht $x > y, z$ für $x > y$ und $x > z$. Entsprechend steht $x, y > z$ für $x > z$ und $y > z$. Für eine Reduktionsordnung $\succ$, einen Term s und eine Folge von Literalen $\Gamma \equiv (t_1 \simeq t_2), \ldots, (t_{2n-1} \simeq t_{2n})$ stehe der Ausdruck $s \succ \Gamma$ für den Vergleich von s mit allen in Γ vorkommenden Termen, also für $s \succ t_1$ und $s \succ t_2$ und $\ldots$ und $s \succ t_{2n}$. Im Folgenden bezeichne $\succ$ eine beliebige, aber feste, grundtotale Reduktionsordnung mit Teiltermeigenschaft und $\succ_l$, $\succ_c$ die zugehörigen Ordnungen auf Literalen und Klauseln. Ist aus dem Kontext offensichtlich, welche Ordnung gemeint ist, wird auch für $\succ_l$ und $\succ_c$ nur kurz $\succ$ geschrieben.

1.6. Constraints

Ein *atomarer Gleichheitsconstraint* ist ein Ausdruck der Form $s \doteq t$, wobei s und t Terme sind. Ein atomarer Gleichheitsconstraint $s \doteq t$ wird von einer Substitution σ *erfüllt*, wenn σ Unifikator der Terme s und t ist. Ein *atomarer Ordnungsconstraint* ist ein Ausdruck der Form $s > t$. Eine Substitution σ erfüllt $s > t$, wenn $s\sigma \succ t\sigma$ gilt. Atomare Gleichheitsconstraints und atomare Ordnungsconstraints werden zu *atomaren Constraints* zusammengefasst. Ein *Constraint* ist eine Menge atomarer Constraints. Eine Substitution erfüllt einen Constraint, wenn sie alle enthaltenen atomaren Constraints erfüllt. Daher wird ein Constraint $\{a_1, a_2, \ldots, a_n\}$ mit den atomaren Constraints $a_1, a_2, \ldots, a_n$ auch in der Form $a_1 \wedge a_2 \wedge \ldots \wedge a_n$ geschrieben. Existiert eine solche erfüllende Substitution, so heißt der Constraint *erfüllbar*, ansonsten *unerfüllbar*. Ein Constraint, der von jeder beliebigen Substitution erfüllt wird, heißt *allgemeingültig*. Da zwischen allgemeingültigen Constraints meist nicht unterschieden werden muss, bezeichne $\top$ einen beliebigen solchen allgemeingültigen Constraint.

Ein Constraint $\aleph$ heißt *schwächer auf einer Variablenmenge* $\mathcal{Y} \subseteq \mathcal{X}$ als ein Constraint $\beth$ (Schreibweise: $\aleph \ll_{\mathcal{Y}} \beth$) genau dann, wenn es für jede $\beth$ erfüllende Substitution σ eine $\aleph$ erfüllende Substitution θ gibt mit $y\sigma \equiv y\theta$ für alle $y \in \mathcal{Y}$. Entsprechend heißt $\beth$ *stärker auf* $\mathcal{Y}$ als $\aleph$ (Schreibweise: $\beth \gg_{\mathcal{Y}} \aleph$). Gilt $\mathcal{Y} = \mathcal{X}$, so heißt $\aleph$ einfach *schwächer* als $\beth$ bzw. $\beth$ stärker als $\aleph$ (Schreibweise: $\aleph \ll \beth$ bzw. $\beth \gg \aleph$). Schließlich heißen $\aleph$ und $\beth$ genau dann *äquivalent auf* $\mathcal{Y}$ bzw. *äquivalent*, wenn $\aleph \ll_{\mathcal{Y}} \beth$ und $\aleph \gg_{\mathcal{Y}} \beth$ bzw. $\aleph \ll \beth$ und $\aleph \gg \beth$ gelten(Schreibweise: $\aleph \equiv_{\mathcal{Y}} \beth$ bzw. $\aleph \equiv \beth$). Beispiel: Der Constraint $\aleph := x \doteq f(z)$ ist schwächer als der Constraint $\beth := y \doteq b,\ g(a, x) \doteq g(a, f(z))$. Es gilt aber $\aleph \equiv_{\{x\}} \beth$.

Die Anwendung einer Substitution σ auf einen atomaren Constraint $s \doteq t$ beziehungsweise $s > t$ ist definiert als $s\sigma \doteq t\sigma$ beziehungsweise $s\sigma > t\sigma$. Entsprechend ist für einen Constraint $\aleph \equiv \aleph_1 \wedge \ldots \wedge \aleph_n$ und eine Substitution σ der Constraint $\aleph\sigma$ definiert als $\aleph_1\sigma \wedge \ldots \wedge \aleph_n\sigma$.

Besteht ein Constraint nur aus atomaren Gleichheitsconstraints, so wird er *reiner Gleichheitsconstraint* genannt; besteht er nur aus atomaren Ordnungsconstraints so wird er *reiner Ordnungsconstraint* genannt. Im Zusammenhang mit Unifikation wird ein reiner Gleichheitsconstraint $\aleph \equiv s_1 \doteq t_1 \wedge \ldots \wedge s_n \doteq t_n$ als Menge von Termpaaren $\{(s_1, t_1), \ldots, (s_n, t_n)\}$ aufgefasst. Insbesondere ist damit $mgu(\aleph)$ gegeben durch den allgemeinsten Unifikator, der s_1 mit t_1, s_2 mit t_2 usw. unifiziert. Für einen Constraint $\aleph$ ist der *Gleichheitsanteil* $eqpart(\aleph)$ definiert durch $eqpart(\aleph) = \{s \doteq t \in \aleph\}$, also die Menge aller Gleichheitsconstraints aus $\aleph$. Der *Ordnungsanteil* eines Constraints $\aleph$ wird definiert als $ordpart(\aleph) = \{s\sigma > t\sigma \mid s > t \in \aleph\}$, wobei $\sigma = mgu(eqpart(\aleph))$ ist.

Eine *Klausel mit Constraint* ist eine Klausel, der ein Constraint zugeordnet ist (Schreibweise: $C \mid \aleph$, wobei C die Klausel und $\aleph$ der Constraint ist). Der Constraint dient dazu, die Instanzen der Klausel zu beschränken. Unter der *leeren Klausel* $\square$ wird im Zusammenhang mit Constraints die Klausel ohne Literale mit einem erfüllbaren Constraint verstanden. Diese Einschränkung ist nötig, damit die leere Klausel unerfüllbar ist. Eine *Instanz* einer Klausel mit Constraint $C \mid \aleph$ ist eine Klausel mit Constraint $C\sigma \mid \aleph\sigma$, wobei σ eine Substitution ist, so dass $\aleph\sigma$ erfüllbar ist. Ist $C\sigma$ hierbei eine Grundklausel, so heißt $C\sigma \mid \aleph\sigma$ *Grundinstanz* von $C \mid \aleph$. Ist $C\sigma \mid \aleph\sigma$ Grundinstanz so kann die Substitution σ zu einer Substitution σ' erweitert werden, so dass für alle $x \in dom(\sigma)$ der Term $\sigma'(x)$ ein Grundterm ist und außerdem $C\sigma \equiv C\sigma'$ und σ' erfüllt $\aleph$ gelten. Im Folgenden werden Grundinstanzen nur mit solchen Substitutionen gebildet, die den Constraint erfüllen; außerdem wird als Schreibvereinfachung ein solcher erfüllter Constraint für Grundinstanzen oft weggelassen, da er die Menge der Instanzen nicht weiter beeinflussen kann. Daher werden Grundinstanzen von Klauseln mit Constraint wie normale Grundklauseln behandelt. Beispiel: Die Klausel mit Constraint $f(x) \simeq b \mid x \doteq g(y)$ besitzt zum Beispiel durch die Substitution $\{x \leftarrow g(y)\}$ die Instanz $f(g(y)) \simeq b \mid g(y) \doteq g(y)$ und durch $\{x \leftarrow g(a), y \leftarrow a\}$ die Grundinstanz $f(g(a)) \simeq b$. Man kann sich also eine Klausel C ohne Constraint auch als Klausel $C \mid \top$ mit allgemeingültigem Constraint vorstellen. Werden im Folgenden Klauseln mit und ohne Constraint gemischt verwendet, so ist immer implizit die Erweiterung der Klauseln ohne Constraint um einen allgemeingültigen Constraint gemeint.

Zwei Klauseln mit Constraint $C \mid \aleph$ und $D \mid \beth$ heißen genau dann *syntaktisch äquivalent* (Schreibweise: $C \mid \aleph \equiv D \mid \beth$), wenn $C \equiv D$ und $\aleph \equiv_{var(C)} \beth$ gilt. Die Klauseln $C \mid \aleph$ und $D \mid \beth$ heißen genau dann *syntaktisch äquivalent bis auf Variablenumbenennung*, wenn es eine Variablenumbenennung σ gibt, so dass $C\sigma \mid \aleph\sigma \equiv D \mid \beth$ gilt. Wie bei Klauseln wird auch $C \mid \aleph \in M$ für Klauseln mit Constraint geschrieben, wenn sich eine zu $C \mid \aleph$ bis auf Variablenumbenennung syntaktisch äquivalente Klausel in der Klauselmenge M befindet. Man beachte, dass zwei Klauseln $C \mid \aleph$ und $D \mid \beth$ genau dann bis auf Variablenumbenennung syntaktisch äquivalent sind, wenn $\{C{\cdot}\sigma \mid \sigma \text{ erfüllt } \aleph\} = \{D{\cdot}\sigma \mid \sigma \text{ erfüllt } \beth\}$ ist. Beispiel: Die Klausel $f(x) \simeq a \mid y \doteq b,\ g(a,x) \doteq g(a, f(z))$ ist syntaktisch äquivalent zu $f(x) \simeq a \mid x \doteq f(z)$ und syntaktisch äquivalent bis auf Variablenumbenennung zu $f(y) \simeq a \mid y \doteq f(z)$.

1.7. Inferenzen und Inferenzsysteme

Nach Einführung der Grundlagen, kann nun das eigentliche Thema, das automatische Theorembeweisen in der reinen Gleichheitslogik erster Stufe, betrachtet werden. Wie in den Grundlagen dargestellt, lässt sich dieses Problem leicht in ein Erfüllbarkeitsproblem transformieren. Gegeben ist also eine Menge von Klauseln und es soll bestimmt werden, ob diese Menge erfüllbar ist. Dazu werden zu dieser Ausgangsmenge immer mehr Klauseln, die aus den bisherigen Klauseln folgen, hinzugenommen, bis ein Widerspruch gefunden wurde oder man sicher sein kann, dass kein Widerspruch existiert. Dieses Vorgehen soll im Folgenden formalisiert werden.

Eine *Inferenz* ist ein Ausdruck der Form

$$\frac{C_1 \quad \dots \quad C_n}{C},$$

wobei $C_1, \dots, C_n, C$ Klauseln sind und $C_1, \dots, C_n$ paarweise variablendisjunkt sind. Die Klauseln $C_1, \dots, C_n$ heißen *Prämissen*, die Klausel C *Konklusion*. Eine Inferenz beschreibt eine Konstruktionsvorschrift für Klauseln. Sind die Prämissen in einer Klauselmenge S vorhanden (evtl. mit umbenannten Variablen) und ist die Konklusion noch nicht in S, heißt die Inferenz *ziehbar in S*. Ist eine Inferenz ziehbar, darf sie *gezogen werden*, das heißt ihre Konklusion darf zur Menge S hinzugenommen werden. Eine Inferenz heißt *korrekt*, wenn jede Interpretation, die alle Prämissen erfüllt, auch die Konklusion erfüllt, also wenn aus den Prämissen die Konklusion folgt.

Eine Menge von Inferenzen heißt *Inferenzsystem*. Eine Klauselmenge S' heißt *hergeleitet* aus einer Klauselmenge S durch ein Inferenzsystems $\mathcal{I}$, wenn S' durch Ziehen von $\mathcal{I}$-Inferenzen aus S entsteht. S heißt *abgeschlossen bezüglich* eines Inferenzsystems $\mathcal{I}$, wenn keine $\mathcal{I}$-Inferenz in S ziehbar ist. Ist eine Klauselmenge S' aus einer Klauselmenge S durch ein Inferenzsystem $\mathcal{I}$ hergeleitet und ist S' abgeschlossen bezüglich $\mathcal{I}$, so heißt S' *Abschluss* von S bezüglich $\mathcal{I}$. Ein Inferenzsystem heißt *korrekt*, wenn jede enthaltene Inferenz korrekt ist. Wenn die leere Klausel $\square$ Element einer aus einer Klauselmenge S durch ein korrektes Inferenzsystem hergeleiteten Klauselmenge ist, so ist S widersprüchlich. Ein Inferenzsystem $\mathcal{I}$ heißt *vollständig*, wenn mit ihm aus jeder widersprüchlichen Klauselmenge die leere Klausel hergeleitet werden kann.

Ziel muss es also sein, korrekte und vollständige Inferenzsysteme $\mathcal{I}$ zu verwenden. Dann kann so lange die Ausgangsmenge S durch Ziehen von Inferenzen zur Klauselmenge S' erweitert werden, bis

- $\square \in S'$ gilt – dann ist S widersprüchlich – oder
- S' abgeschlossen ist bezüglich $\mathcal{I}$ und $\square \notin S'$ – dann ist S erfüllbar.

Da das Erfüllbarkeitsproblem nicht entscheidbar, sondern nur rekursiv aufzählbar ist, kann nicht garantiert werden, dass nach endlicher Zeit einer der beiden Fälle eintritt. Wird jedoch die *Fairnessbedingung* eingehalten, dass jede ziehbare Inferenz nach endlicher Zeit gezogen wird, so tritt der Fall $\square \in S'$ nach endlicher Zeit ein, falls S widersprüchlich ist.

Die Korrektheit eines Inferenzsystems $\mathcal{I}$ ist leicht nachzuweisen, indem die Korrektheit jeder Inferenz gezeigt wird. Die Vollständigkeit ist schwieriger zu zeigen. Es ist zu zeigen, dass jede Klauselmenge S, deren Abschluss S' bezüglich $\mathcal{I}$ nicht $\square$ enthält, erfüllbar ist. Um dies zu zeigen, wird im Folgenden aus S' ein Modell für S konstruiert. Dieses Modell wird in Form eines konvergenten Reduktionssystems angegeben werden, das, wie oben dargelegt, einem Herbrand-Modell entspricht.

2. Der Superpositionskalkül

2.1. Grundlage: der Paramodulationskalkül

Die hier betrachteten Inferenzsysteme sollen – wie oben motiviert – eine Klauselmenge solange um neues Wissen, das heißt neue Klauseln anreichern, bis offensichtlich wird, ob sie unerfüllbar ist. Die Idee des hier betrachteten Inferenzsystem, der Paramodulation, ist einfach: „Ersetze Gleiches durch Gleiches“. Weiß man, dass $t[l]_p \simeq s$ und $l \simeq r$ gilt, so kann man als neues Wissen $t[r]_p \simeq s$ hinzunehmen. In Inferenzschreibweise ergibt sich für Grundklauseln:

$$\frac{C \vee s \simeq t \qquad D}{C \vee D[t]_p}$$

falls

- $s \equiv D|_p$

Betrachtet man eine allgemeine Klausel als Repräsentation aller ihrer Grundinstanzen, so ergibt sich damit für allgemeine Klauseln:

$$\frac{C \vee s \simeq t \qquad D}{(C \vee D[t]_p)\sigma}$$

falls

- $\sigma = mgu(s, D|_p)$

Man sieht jedoch, dass die erzeugten Klauseln – da sie Multimengen von Literalen sind – immer mindestens so viele Literale besitzen wie die Klauseln, aus denen sie entstehen. Um auch die leere Klausel erzeugen zu können, ist es daher nötig, offensichtlich nicht erfüllbare Literale, das heißt Literale der Form $s \not\simeq s$ entfernen zu dürfen. Für allgemeine Klauseln ergibt sich damit folgende Inferenz

$$\frac{\Gamma, s \simeq t \rightarrow \Delta}{(\Gamma \rightarrow \Delta)\sigma}$$

falls

- $\sigma = mgu(s, t)$

Beim Anwenden dieser beiden Inferenzen wird ohne weitere Einschränkungen der Suchraum jedoch so groß, dass es in der Praxis sehr lange dauert, einen Widerspruch zu finden. Daher ist es sinnvoll, den Suchraum durch Verfeinerungen dieser Inferenzen einzuschränken. Bekannte nützliche Einschränkungen des Suchraums sind:

1. Es werden keine Inferenzen an Variablenpositionen gezogen, also darf $D|_p$ keine Variable sein.

2. Es wird eine beliebige, aber feste Reduktionsordnung $\succ$ verwendet und Inferenzen werden nur mit maximalen Seiten und in maximale Seiten maximaler Literale bezüglich dieser Reduktionsordnung gezogen.

3. Bei der Superposition werden Regeln nicht auf Literale angewendet, die kleiner sind als die Regel.

Die eingeführten Inferenzregeln dürfen nur unter den angegebenen Bedingung gezogen werden. Diese Bedingungen sind notwendig für die Korrektheit der Inferenzregeln und dürfen keinesfalls vernachlässigt werden. Die Einschränkungen des Suchraums werden ebenfalls über Bedingungen realisiert. Diese Bedingungen sind jedoch nicht für die Korrektheit der Inferenzen nötig, sondern dienen nur einer beschleunigten Beweisfindung. Sie können daher auch vernachlässigt werden.

Man sollte also zwischen diesen beiden Arten von Bedingungen unterscheiden. Im Folgenden werden Bedingungen, die für die Korrektheit nötig sind, in der Aufzählung mit dem Zeichen $\bullet$ gekennzeichnet und Bedingungen, die nur der Suchraumeinschränkung dienen und nicht für die Korrektheit notwendig sind, mit dem Zeichen $\circ$ gekennzeichnet.

Erweitert man die Paramodulation um die genannten Einschränkungen des Suchraums, so entsteht die sogenannte *Superposition*. Im Folgenden soll der Superpositionskalkül schrittweise eingeführt werden. Begonnen wird mit dem Superpositionskalkül für Grundhornklauseln. An diesem einfachen Fall sollen die grundlegenden Ideen vermittelt werden. Aufbauend darauf wird der Superpositionskalkül immer weiter verfeinert.

2.2. Der Superpositionskalkül für Grundhornklauseln

Das Inferenzsystem $\mathcal{G}$ für Grundhornklauseln besteht aus folgenden Inferenzregeln:

Superposition rechts:

$$\frac{\Gamma' \to l \simeq r \qquad \Gamma \to s \simeq t}{\Gamma', \Gamma \to s[r]_p \simeq t}$$

falls

- $\bullet$ $s|_p \equiv l$
- $\circ$ $l \succ r, \Gamma'$
- $\circ$ $s \succ t, \Gamma$
- $\circ$ $s \simeq t \succ l \simeq r$

Superposition links:

$$\frac{\Gamma' \to l \simeq r \qquad \Gamma, s \simeq t \to \Delta}{\Gamma', \Gamma, s[r]_p \simeq t \to \Delta}$$

falls

- $\bullet$ $s|_p \equiv l$
- $\circ$ $l \succ r, \Gamma'$
- $\circ$ $s \succ t$ und $s \succeq \Gamma, \Delta$

Reflexivitäts-Resolution:

$$\frac{\Gamma, s \simeq s \to \Delta}{\Gamma \to \Delta}$$

falls

- $\circ$ $s \succeq \Gamma, \Delta$

Satz 1 Sei S der Abschluss einer Menge S_0 von Grundhornklauseln bezüglich $\mathcal{G}$. Dann ist S_0 genau dann erfüllbar, wenn S nicht die leere Klausel enthält.

Beweis Ist die leere Klausel in S enthalten, so ist S_0 unerfüllbar, da die Inferenzen korrekt sind.

Ist die leere Klausel nicht in S enthalten, ist zu zeigen, dass S_0 erfüllbar ist. Es ist also zu zeigen, dass ein Modell von S_0 existiert. Wenn ein Modell von S_0 existiert, so existiert auch ein Herbrand-Modell von S_0. Im Folgenden wird ein solches Herbrand-Modell konstruiert werden. Dieses Herbrand-Modell wird in Form eines Reduktionssystems R angegeben, das mittels Induktion über $\succ_c$ wie folgt definiert wird:

Eine Klausel $C \equiv \Gamma \to l \simeq r$ *generiert die Regel* $l \Rightarrow r$ genau dann, wenn gilt:

1. $C \in S$
2. $R_C^* \not\models C$
3. $l \succ r, \Gamma$
4. l ist nicht reduzierbar bezüglich R_C

wobei R_C die Menge der von Klauseln $D \in S$ mit $D \prec_c C$ generierten Regeln ist. Unter R wird die Menge der von allen Klauseln aus S generierten Regeln verstanden. R^C schließlich sei R_C vereinigt mit der evtl. von C generierten Regel. Eine Klausel, die eine Regel generiert, heißt *generierend*.

Lemma 1 Alle Reduktionsysteme R' mit $R' \subseteq R$ sind konvergent.

Beweis Da für alle generierten Regeln $l \Rightarrow r$ die Beziehung $l \succ r$ gilt und $\succ$ eine Reduktionsordnung ist, ist R' terminierend.

Um die Konfluenz zu zeigen, genügt es also, die lokale Konfluenz zu zeigen. Diese ist im Grundfall gegeben, wenn keine linke Seite l' einer Regel ein Teilterm einer anderen linken Seite l ist.

Angenommen es gibt zwei Regeln $l \Rightarrow r$ und $l' \Rightarrow r'$ aus R', so dass l' Teilterm von l ist. Die Regel $l \Rightarrow r$ werde von der Klausel C generiert; $l' \Rightarrow r'$ von der Klausel C'. Da C die Regel $l \Rightarrow r$ generiert, ist durch Bedingung 4 garantiert, dass $l' \Rightarrow r' \notin R_C$ gilt. Da nach Voraussetzung nicht $C' \equiv C$ gelten kann, muss also $C' \succ_c C$ gelten. Nach Definition von $\succ_c$ und wegen Bedingung 3 gilt dann $l' \succeq l$. Gilt $l' \equiv l$, so ist l' wegen $l \Rightarrow r \in R_{C'}$ reduzierbar bezüglich $R_{C'}$. Dies ist aber ein Widerspruch dazu, dass C' generierend ist. Also muss $l' \succ l$ gelten. Dies ist aber ein Widerspruch dazu, dass l' ein Teilterm von l ist. Also gibt es in R' keine solchen Regeln $l \Rightarrow r$ und $l' \Rightarrow r'$.

Das Reduktionssystem R' ist also auch konfluent und daher konvergent. □

Lemma 2 Sei C eine Grundhornklausel. Dann gilt für alle Regeln $l \Rightarrow r \in R \setminus R_C$ und alle Teilterme s von C die Beziehung $l \succeq s$. Gilt für einen Teilterm $s \equiv l$, so besitzt C die Form $\Gamma \to s \simeq t$ mit $s \succ t, \Gamma$.

Beweis Sei also $l \Rightarrow r$ eine beliebige Regel aus $R \setminus R_C$ und s ein beliebiger Teilterm von C. Weiter sei $C' \in S$ die Klausel, die $l \Rightarrow r$ generiert. Dann ist l maximal in C'

und wegen $l \Rightarrow r \in R \setminus R_C$ gilt $C' \succeq_c C$. Nach Definition von $\succ_c$ gilt somit $l \succeq s$, da s ein Teilterm von C ist.

Es gelte $l \equiv s$. Weil C' generierend ist, ist l strikt maximal in C' und kommt somit in C' nur in dem positiven Literal vor. Dann muss s strikt maximal in C sein und in dem einzigen positiven Literal vorkommen, da ansonsten $C \succ C'$ gelten würde. Damit muss C die Form $\Gamma \rightarrow s \simeq t$ mit $s \succ \Gamma, t$ besitzen. □

Lemma 3 Sei $C \in S$ eine Grundhornklausel. Dann gilt für alle Regeln $l \Rightarrow r \in R \setminus R^C$ und alle Teilterme s von C die Beziehung $l \succeq s$. Gilt für einen Teilterm $s \equiv l$, so besitzt C die Form $\Gamma \rightarrow s \simeq t$ mit $s \succ t, \Gamma$ und $R_C^* \not\models \Gamma$.

Beweis Wegen $R \setminus R^C \subseteq R \setminus R_C$ folgt aus Lemma 2, dass für alle Regeln $l \Rightarrow r \in R \setminus R^C$ und alle Teilterme s von C die Beziehung $l \succeq s$ gilt. Weiter folgt, dass wenn für einen Teilterm $s \equiv l$ gilt, C die Form $\Gamma \rightarrow s \simeq t$ mit $s \succ t, \Gamma$ besitzt. Zu zeigen bleibt, dass in diesem Fall auch $R_C^* \not\models \Gamma$ gilt. Sei C' die Klausel, die die Regel $l \Rightarrow r$ generiert. Wegen $l \Rightarrow r \in R \setminus R^C$ gilt $C' \succ C$.

Angenommen es gilt $R_C^* \not\models C$. Damit ist entweder C generierend oder es gibt eine Regel in R_C, durch die $s \equiv l$ reduzierbar ist. Damit ist aber l reduzierbar in $R_{C'} \supseteq R^C$ und damit kann C' nicht generierend sein. Also muss die Annahme $R_C^* \not\models C$ falsch sein.

Es gilt also $R_C^* \models C \equiv \Gamma \rightarrow l \simeq r$. Daher gilt $R_C^* \models s \simeq t$ oder $R_C^* \not\models \Gamma$. Angenommen es gilt $R_C^* \models s \simeq t$. Wegen $s \succ t$ muss somit s reduzierbar bezüglich R_C sein. Aber wegen $R_C \subseteq R_{C'}$ ist dann auch $l \equiv s$ reduzierbar bezüglich $R_{C'}$. Dies ist ein Widerspruch dazu, dass C' generierend ist und somit gilt $R_C^* \not\models s \simeq t$. Dann muss aber $R_C^* \not\models \Gamma$ gelten. □

Lemma 4 Sei C eine Grundhornklausel der Form $\Gamma, \Gamma' \rightarrow \Delta$. Dann gilt für alle Grundhornklauseln D mit $D \succ C$: $R_C^* \models \Gamma$ gdw. $R^{C^*} \models \Gamma$ gdw. $R_D^* \models \Gamma$ gdw. $R^* \models \Gamma$.

Beweis Gilt $R_C^* \models \Gamma$, so gilt auch $R^{C^*} \models \Gamma$, $R_D^* \models \Gamma$ und $R^* \models \Gamma$, da $R_C \subseteq R^C \subseteq R_D \subseteq R$ gilt und Γ nur Gleichungen enthält.

Also gelte $R_C^* \not\models \Gamma$. Dann existiert eine Gleichung $u \simeq v$ in Γ mit $R_C^* \not\models u \simeq v$. Seien u', v' die Normalformen zu u und v bezüglich R_C. Dann gilt $u' \not\equiv v'$. Sei $l \Rightarrow r \in R \setminus R_C$, also eine beliebige später hinzukommende Regel. Nach Lemma 2 gilt $l \succ u$ und $l \succ v$, da u und v in einem negativen Literal vorkommen. Also gilt $l \succ u \succeq u'$ und $l \succ v \succeq v'$. Damit sind weder u' noch v' reduzierbar durch die Regel $l \Rightarrow r$.

Da u' und v' somit bezüglich keiner Regel aus $R \setminus R_C$ und nach Voraussetzung auch bezüglich keiner Regel aus R_C reduzierbar sind, sind u' und v' auch Normalformen bezüglich R. Da R nach Lemma 1 konvergent ist, gilt also $R^* \not\models u \simeq v$. Somit gilt $R^* \not\models \Gamma$ und wegen $R^C \subseteq R_D \subseteq R$ gelten auch $R^{C^*} \not\models \Gamma$ und $R_D^* \not\models \Gamma$. □

Folgerung 1 (Folgerung aus Lemma 4) Für zwei Grundhornklauseln C und D der Form $C \equiv \Gamma, \Gamma' \rightarrow \Delta$ und $D \equiv \Gamma, \Gamma'' \rightarrow \Delta'$ gilt $R_C^* \models \Gamma$ gdw. $R_D^* \models \Gamma$.

Lemma 5 Sei C eine Grundhornklausel. Dann gilt für alle Grundhornklauseln D mit $D \succ C$: gilt $R^* \not\models C$, so gilt auch $R_C^* \not\models C$, $R^{C^*} \not\models C$ und $R_D^* \not\models C$.

Beweis C besitze die Form $\Gamma \rightarrow \Delta$. Da C eine Grundhornklausel ist, enthält dabei Δ höchstens eine Gleichung, kann jedoch auch leer sein.

Es gilt $R^* \not\models C$ und daher gelten $R^* \models \Gamma$ und $R^* \not\models \Delta$. Nach Lemma 4 gelten dann auch $R_D^* \models \Gamma$, $R^{C^*} \models \Gamma$ und $R_C^* \models \Gamma$. Da Δ nur aus Gleichungen besteht und wegen $R_C \subseteq R^C \subseteq R_D \subseteq R$, gilt dann auch $R_C^* \not\models \Delta$, $R^{C^*} \not\models \Delta$ und $R_D^* \not\models \Delta$. Insgesamt ergibt sich also $R_D^* \not\models C$, $R^{C^*} \not\models C$ und $R_C^* \not\models C$. $\square$

Lemma 6 Sei $C \in S$ eine Grundhornklausel. Dann gilt für alle Grundhornklauseln D mit $D \succ C$: gilt $R^{C^*} \not\models C$ so gelten auch $R_D^* \not\models C$ und $R^* \not\models C$.

Beweis C besitze die Form $\Gamma \rightarrow \Delta$. Da C eine Grundhornklausel ist, enthält dabei Δ höchstens eine Gleichung, kann jedoch auch leer sein.

Da $R^{C^*} \not\models C$ gilt, müssen $R^{C^*} \models \Gamma$ und $R^{C^*} \not\models \Delta$ gelten. Nach Lemma 4 gilt dann auch $R_D^* \models \Gamma$ und $R^* \models \Gamma$. Zu zeigen bleibt $R_D^* \not\models \Delta$ sowie $R^* \not\models \Delta$.

Ist Δ leer, enthält also C keine positiven Literale, so ist nichts zu zeigen. Da C eine Hornklausel ist, bestehe Δ also aus genau einem positiven Literal $u \simeq v$. Wegen $R^{C^*} \not\models \Delta$ gilt $R^{C^*} \not\models u \simeq v$. Seien u', v' die Normalformen zu u und v bezüglich R^C. Dann gilt $u' \not\equiv v'$. Sei $l \Rightarrow r \in R \setminus R^C$, also eine beliebige später hinzukommende Regel. Wegen $R^{C^*} \models \Gamma$ gilt nach Lemma 4 $R_C^* \models \Gamma$. Damit gilt nach Lemma 3 dann $l \succ u \succeq u'$ und $l \succ v \succeq v'$. Also sind weder u' noch v' reduzierbar durch die Regel $l \Rightarrow r$. Da u' und v' somit bezüglich keiner Regel aus $R \setminus R^C$ und nach Voraussetzung auch bezüglich keiner Regel aus R^C reduzierbar sind, sind u' und v' auch Normalformen bezüglich R. Da R nach Lemma 1 konvergent ist, gilt also $R^* \not\models u \simeq v$. Somit gilt $R^* \not\models \Delta$ und wegen $R_D \subseteq R$ gilt auch $R_D^* \not\models \Delta$. Insgesamt gilt also $R^* \not\models C$ und $R_D \not\models C$. $\square$

Folgerung 2 (Folgerung aus Lemma 5 und Lemma 6) Sei $C \in S$ eine Grundhornklausel. Dann gilt für alle Grundhornklauseln D mit $D \succ C$: $R^{C^*} \models C$ gdw. $R_D^* \models C$ gdw. $R^* \models C$.

Beweis $R^{C^*} \models C$ gdw. $R^* \models C$ sowie die Aussage, dass, wenn $R^{C^*} \not\models C$ gilt, auch $R_D^* \not\models C$ gilt, folgen direkt aus Lemma 5 und Lemma 6. Die noch fehlende Aussage, dass, wenn $R^{C^*} \models C$ gilt, auch $R_D^* \models C$ gilt, ergibt sich aus Lemma 4 und $R^C \subseteq R_D$. $\square$

Zu zeigen bleibt, dass R ein Modell von S_0 ist. Dazu wird per Induktion über die Klauselordnung $\succ_c$ gezeigt, dass R ein Modell von $S \supseteq S_0$ ist, da es kein minimales *Gegenbeispiel* gibt, das heißt keine minimale Klausel $C \in S$ mit $R^* \not\models C$. Bei der Induktion ist zu beachten, dass S eine Menge von Grundklauseln und $\succ_c$ grundtotal ist. Damit sind alle Klauseln miteinander vergleichbar.

Angenommen R ist kein Modell von S. Dann gibt es ein Gegenbeispiel und damit auch ein minimales Gegenbeispiel $C \in S$. Für C gilt also $R^* \not\models C$ und für alle anderen $D \in S$ mit $R^* \not\models D$ gilt $D \not\prec C$. Da $\succ$ grundtotal ist und C und D Grundklauseln sind, gilt sogar $D \succ C$.

Dann kann man abhängig vom Vorkommen des maximalen Terms s in C folgende vier Fälle unterscheiden:

- Fall 1: C besitzt die Form $\Gamma \rightarrow s \simeq s$.

  Dann ist C eine Tautologie, was ein Widerspruch zur Voraussetzung $R^* \not\models C$ ist.

- Fall 2: C besitzt die Form $\Gamma \rightarrow s \simeq t$ mit $s \not\equiv t$ und der maximale Term s kommt nicht in Γ vor.

  Nach Voraussetzung gilt $s \succ t, \Gamma$. Weiter muss wegen $R^* \not\models C$ nach Folgerung 2 $R^*_C \not\models C$ gelten. Wegen $R^* \not\models C$ kann C aber nicht generierend sein. Daher muss s mit einer Regel $l \Rightarrow r \in R_C$ an einer Position p reduzierbar sein.

  Sei $C' \equiv \Gamma' \rightarrow l \simeq r \in S$ die Klausel, die $l \Rightarrow r$ generiert. Dann gilt wegen $l \Rightarrow r \in R_C$ die Beziehung $C \succ C'$. Da $l \simeq r$ das maximale Literal in C' und $s \simeq t$ das maximale Literal in C ist, gilt daher $s \simeq t \succeq l \simeq r$. Angenommen es würde $s \simeq t \equiv l \simeq r$ gelten, dann würde aber $s \simeq t \in R^{C'} \subseteq R$ und damit $R^* \models C$ gelten. Also gilt $s \simeq t \succ l \simeq r$ und somit ist folgende Inferenz vom Typ *Superposition rechts* ziehbar:

$$\frac{\Gamma' \rightarrow l \simeq r \qquad \Gamma \rightarrow s \simeq t}{\Gamma', \Gamma \rightarrow s[r]_p \simeq t}$$

  Für die Konklusion D gilt $D \prec C$, denn es gilt $s \equiv s[l]_p \succ s[r]_p$ (wegen $l \succ r$) und $s \succeq l \succ \Gamma'$. Außerdem ist $D \in S$, da S abgeschlossen unter $\mathcal{G}$ ist. Es gilt $R^* \not\models D$, denn:

  - $R^* \models \Gamma$ wegen $R^* \not\models C$
  - Da C' generierend ist, gilt $R^*_{C'} \not\models C'$ und somit $R^* \supseteq R^*_{C'} \models \Gamma'$.
  - $R^* \not\models s[r]_p \simeq t$, da wegen $l \Rightarrow r \in R$ sonst auch $R^* \models s[l]_p \simeq t$ und damit $R^* \models C$ gelten würde.

  Also ist D ein kleineres Gegenbeispiel als C. Dies ist aber ein Widerspruch zur Minimalität von C.

- Fall 3: C besitzt die Form $\Gamma, s \simeq s \rightarrow \Delta$.

  Dann ist folgende Inferenz vom Typ *Reflexivitäts-Resolution* ziehbar:

$$\frac{\Gamma, s \simeq s \rightarrow \Delta}{\Gamma \rightarrow \Delta}$$

  Die Konklusion D liegt in S, da S abgeschlossen unter $\mathcal{G}$ ist. Außerdem gilt $R^* \not\models D$, da $R^* \not\models C$. Wegen $C \succ D$ ist dies ein Widerspruch zur Minimalität von C.

- Fall 4: C besitzt die Form $\Gamma, s \simeq t \rightarrow \Delta$ mit $s \succ t$.

  Wegen $R^* \not\models C$ gilt $R^* \models s \simeq t$. Somit müssen s und t die gleiche Normalform bezüglich R besitzen. Also muss s an einer Position p reduzierbar durch eine

Regel $l \Rightarrow r \in R$ sein. Angenommen $l \Rightarrow r$ wurde von $C' := \Gamma' \to l \simeq r \in S$ generiert. Dann ist folgende Inferenz vom Typ *Superposition links* ziehbar:

$$\frac{\Gamma' \to l \simeq r \qquad \Gamma, s \simeq t \to \Delta}{\Gamma', \Gamma, s[r]_p \simeq t \to \Delta}$$

Es gilt $s \succ s[r]_p$ und wegen $s \succeq l \succ \Gamma'$ auch $s \succ \Gamma'$. Damit gilt für die Konklusion D die Beziehung $D \prec C$. Außerdem ist $D \in S$, da S abgeschlossen unter $\mathcal{G}$ ist. Es gilt $R^* \not\models D$, denn

- da C' generierend ist, gilt $R^* \supseteq R^*_{C'} \models \Gamma'$
- $R^* \models s[r]_p \simeq t$, da $R^* \models s[l]_p \simeq t,\ l \simeq r$
- $R^* \models \Gamma$ wegen $R^* \not\models C$
- $R^* \not\models \Delta$ wegen $R^* \not\models C$

Also ist D ein kleineres Gegenbeispiel als C. Dies ist aber ein Widerspruch zur Minimalität von C.

Da nach Voraussetzung S nicht die leere Klausel enthält und somit $C \not\equiv \Box$ gilt und C eine Grundhornklausel ist, sind dies alle möglichen Fälle. Jeder dieser Fälle führt zu einem Widerspruch. Also gibt es kein minimales Gegenbeispiel. Für alle $D \in S$ gilt also $R^* \models D$. Damit ist R ein Modell und $S \supseteq S_0$. Also ist S_0 erfüllbar. $\Box$

Die Einführung des Inferenzsystems $\mathcal{G}$ dient hauptsächlich der Darstellung der grundlegenden Ideen. Vor allem die Modellkonstruktion und damit die Beweisidee zu Satz 1 sind wichtig und werden im Folgenden immer wieder benutzt und dabei verfeinert werden.

2.3. Suchraumeinschränkung durch Selektion

Eine weitere Methode, den Suchraum zu verkleinern, ist die sogenannte *Selektion*. Man erlaubt in den Klauseln der Ausgangsmenge und nach dem Ziehen einer Inferenz in deren Konklusion willkürlich negative Literale zu selektieren, das heißt besonders zu kennzeichnen (im Folgenden durch Unterstreichung gekennzeichnet). Anschließend werden eine Klausel mit einem selektieren Literal betreffende Inferenzen nur noch gezogen, wenn sie das selektierte Literal betreffen. Es wird davon ausgegangen, dass sich durch Anwenden einer Substitution die Selektiertheit einer Klausel nicht ändert. Dies ist zwar für das folgende Inferenzsystem unerheblich, da dieses nur auf Grundklauseln arbeitet, wird jedoch später benötigt.

Um durch diese Einschränkung die Vollständigkeit des Inferenzsystems nicht zu verlieren ist es notwendig, folgende *Selektionsregel* zu beachten: Kommt in einer Klausel $\Gamma \to \Delta$ der maximale Term in Γ vor, so muss in dieser Klausel ein Literal selektiert sein.

Erweitert man das Inferenzsystem $\mathcal{G}$ um diese Technik, so ergibt sich das folgende Inferenzsystem $\mathcal{G}$ mit Selektion:

Superposition rechts:

$$\frac{\Gamma' \to l \simeq r \qquad \Gamma \to s \simeq t}{\Gamma', \Gamma \to s[r]_p \simeq t}$$

falls

- $s|_p \equiv l$
- ◦ $l \succ r, \Gamma'$
- ◦ $s \succ t, \Gamma$
- ◦ $s \simeq t \succ l \simeq r$
- ◦ in Γ, Γ' ist keine Gleichung selektiert

Superposition links:

$$\frac{\Gamma' \to l \simeq r \qquad \Gamma, \underline{s \simeq t} \to \Delta}{\Gamma', \Gamma, s[r]_p \simeq t \to \Delta}$$

falls

- $s|_p \equiv l$
- ◦ $l \succ r, \Gamma'$
- ◦ $s \succ t$
- ◦ in Γ' ist keine Gleichung selektiert

Reflexivitäts-Resolution:

$$\frac{\Gamma, \underline{s \simeq s} \to \Delta}{\Gamma \to \Delta}$$

Die Korrektheit dieses Inferenzsystems ergibt sich erneut aus der Korrektheit der einzelnen Inferenzen und kann leicht nachgewiesen werden.

Um die Vollständigkeit des Inferenzsystems zu beweisen, kann man den Beweis aus Abschnitt 2.2 anpassen. Dabei wird vorausgesetzt, dass die oben genannte Selektionsregel beachtet wird. Essenziell für den Vollständigkeitsbeweis ist das folgende Lemma:

Lemma 7 Keine Klausel $C \in S$ mit einer selektierten Gleichung ist generierend.

Beweis Angenommen die Behauptung gilt nicht. Dann gibt es eine generierende Klausel $C \in S$ mit einer selektierten Gleichung. Sei $C \equiv \Gamma, \underline{u \simeq v} \to s \simeq t$ die kleinste solche Klausel.

- Fall 1: $u \equiv v$

  Es kann eine Inferenz vom Typ *Reflexivitäts-Resolution* gezogen werden:

  $$\frac{\Gamma, \underline{u \simeq v} \to s \simeq t}{\Gamma \to s \simeq t}$$

  Da S abgeschlossen ist bezüglich des Inferenzsystems $\mathcal{G}$ mit Selektion, ist die Konklusion $D \equiv \Gamma \to s \simeq t \in S$. Es gilt $D \prec C$ und somit $R_D \subseteq R_C$. Außerdem ist die Klausel D generierend, denn:

  1. Es gilt $R_D^* \not\models D$, denn:
     $R_D^* \models \Gamma$ nach Folgerung 1, da $R_C^* \models \Gamma$. Wegen $R_C^* \not\models C$ und $R_D \subseteq R_C$ gilt $R_D^* \not\models s \simeq t$. Damit gilt $R_D^* \not\models D$.

2. Es gilt $s \succ t, \Gamma$, da C generierend ist
3. Der Term s ist nicht reduzierbar bezüglich R_D, da s nicht reduzierbar bezüglich $R_C \supseteq R_D$ ist.

Also ist D generierend und generiert $s \Rightarrow t$. Da $C \succ D$, gilt damit aber $R_C^* \models s \simeq t$ und somit $R_C^* \models C$. Dies ist ein Widerspruch dazu, dass C generierend ist.

- Fall 2: $u \not\equiv v$

  Sei o. B. d. A. $u \succ v$. Da C generierend ist, gilt $R_C^* \not\models C$ und somit $u \Downarrow_{R_C} v$. Daher gibt es eine Regel $l \Rightarrow r \in R_C$ und eine Position p mit $u \equiv u[l]_p \Rightarrow_{R_C} u[r]_p \Downarrow_{R_C} v$. Sei $C' := \Gamma' \rightarrow l \simeq r$ die Klausel aus S, die $l \Rightarrow r$ generiert. Somit ist C' eine generierende Klausel mit $C' \prec C$.

  Dann kann diese Inferenz vom Typ *Superposition links*

$$\frac{\Gamma' \rightarrow l \simeq r \qquad \Gamma, \underline{u \simeq v} \rightarrow s \simeq t}{\Gamma', \Gamma, u[r]_p \simeq v \rightarrow s \simeq t}$$

  gezogen werden, denn es gilt:

  - $u|_p \equiv l$ nach Voraussetzung
  - $l \succ r, \Gamma'$, da C' generierend ist
  - $u \succ v$ nach Voraussetzung
  - In Γ' ist keine Gleichung selektiert, da C nach Voraussetzung die kleinste generierende Klausel ist, die eine selektierte Gleichung enthält und C' eine generierende Klausel mit $C' \prec C$ ist.

  Da S abgeschlossen bezüglich des Inferenzsystems $\mathcal{G}$ mit Selektion ist, ist die Konklusion $D := \Gamma', \Gamma, u[r]_p \simeq v \rightarrow s \simeq t \in S$. Es gilt $D \prec C$ (wegen $u \succ u[r]_p$ und $u \succeq l \succ \Gamma'$), aber $D \succ C'$, da für den maximalen Term l aus D gilt $l \preceq u$ und $u \prec s$ gilt, da C generierend ist. Außerdem ist D generierend, denn

  1. s ist strikt maximal in D, denn
     - $s \succ t, v, \Gamma$, da C generierend ist
     - Es gilt $s \succ u$, da C generierend ist. Wegen $l \succ r$ gilt weiter $u \succ u[r]_p$ und somit $s \succ u[r]_p$.
     - Da C' generierend ist, gilt $l \succ \Gamma'$, woraus zusammen mit $s \succ u \succeq l$ folgt $s \succ \Gamma'$.
  2. Es gilt $R_D^* \not\models D$, denn
     - $R_D^* \models \Gamma$ gilt wegen $R_C^* \models \Gamma$ nach Folgerung 1
     - $R_D^* \models \Gamma'$ gilt wegen $R_{C'}^* \models \Gamma'$ nach Folgerung 1
     - Es gilt $R_C^* \models u \simeq v$. Da aber nach Lemma 2 für alle Regeln $l' \Rightarrow r'$ aus $R \backslash R_D$ und damit auch für alle Regeln aus $R_C \backslash R_D$ die Beziehung $l' \succeq s \succ u \succ v$ gilt, muss bereits in $R_D \subseteq R_C$ gelten $R_D^* \models u \simeq v$. Weiterhin gilt $l \Rightarrow r \in R_D$, da $C' \prec D$. Zusammen mit $R_D^* \models u \simeq v$ folgt $R_D^* \models u[r]_p \simeq v$.
     - $R_D^* \not\models s \simeq t$, da nach Voraussetzung $R_C^* \not\models s \simeq t$ und $R_C \supseteq R_D$

3. s ist nicht reduzierbar bezüglich R_D, da s nicht reduzierbar bezüglich $R_C \supseteq R_D$ ist.

Also ist D generierend und generiert $s \Rightarrow t$. Wegen $C \succ D$ gilt damit aber $R_C^* \models s \simeq t$ und somit $R_C^* \models C$. Dies ist ein Widerspruch dazu, dass C generierend ist. □

Mit Hilfe des letzten Lemmas lässt sich im Beweis zu Satz 1 sicherstellen, dass in der linken Prämisse der in Fall 2 verwendeten Superpositions-Inferenzen niemals selektierte Gleichungen vorkommen, da diese stets generierende Klauseln sind. Weiterhin muss die Voraussetzung ausgenutzt werden, dass in jeder Klausel $\Gamma \to \Delta$, in der der maximale Term in Γ vorkommt, ein Literal selektiert ist, um in Fall 3 und 4 sicherzustellen, dass ein passendes Literal des minimalen Gegenbeispiels selektiert ist.

2.4. Der Superpositionskalkül für allgemeine Hornklauseln

Bisher wurden nur Grundhornklauseln behandelt, also Hornklauseln, die keine Variablen enthalten. Nun soll der Superpositionskalkül so verallgemeinert werden, dass auch allgemeine Hornklauseln behandelt werden können. Eine allgemeine Hornklausel repräsentiert alle ihre Grundinstanzen, das heißt eine Menge von Grundhornklauseln. Die Grundidee besteht darin, alle Grundinstanzen aufzuzählen und dann die bekannten Inferenzen aus $\mathcal{G}$ anzuwenden. Dies ist in der Praxis jedoch nicht realisierbar, da die Menge der Grundinstanzen einer Klausel meist unendlich ist. Daher ist es nötig die Inferenzen auf allgemeine Hornklauseln zu *liften*, das heißt die Inferenzen so abzuändern, dass sie direkt mit allgemeinen Hornklauseln benutzt werden können und das Ziehen einer solchen Inferenz eine große Zahl an auf Grundinstanzen ziehbaren Inferenzen abdeckt.

Wurde bisher in den Inferenzen im Grundfall gefordert, dass zwei Grundterme $s\theta$ und $t\theta$ syntaktisch äquivalent sind, so entsteht daraus eine geliftete Inferenz, in deren Konklusion s und t von jeder Grundsubstitution der Konklusion unifiziert werden. Dies kann man erreichen, indem man den allgemeinsten Unifikator $\sigma = mgu(s, t)$ dieser beiden Terme auf die Konklusion anwendet. Dadurch wird die Menge der Grundinstanzen der Konklusion genau auf die Menge der von Unifikatoren von s und t gebildeten Grundinstanzen eingeschränkt. Damit sind bereits alle für die Korrektheit nötigen Bedingungen geliftet. Es können auch die nur der Beschränkung des Suchraums dienenden Ordnungsbedingungen teilweise geliftet werden. Müssen nur die Grundinstanzen betrachtet werden, für die $s\theta \succ t\theta$ gilt, so genügt es, die geliftete Inferenz zu ziehen, wenn $t \not\succeq s$ gilt. Denn gilt $t \succeq s$, so gilt auch $t\theta \succeq s\theta$ für alle Substitutionen θ, da $\succ$ verträglich mit Substitutionen ist. Damit gilt dann aber $s\theta \not\succ t\theta$. Weiß man aufgrund anderer Bedingungen, dass jede Grundinstanz, die betrachtet werden muss, durch eine Substitution entsteht, die Instanz einer Substitution σ ist, so kann man statt $t \not\succeq s$ auch die stärkere Bedingung $t\sigma \not\succeq s\sigma$ überprüfen. Bei dieser Abschätzung $t\sigma \not\succeq s\sigma$ und noch mehr bei $t \not\succeq s$ werden jedoch immer noch Grundinferenzen gezogen, für die $s\theta \succ t\theta$ nicht gilt!

Daraus ergibt sich dann das folgende Inferenzsystem $\mathcal{H}$:

Superposition rechts:

$$\frac{\Gamma' \rightarrow l \simeq r \qquad \Gamma \rightarrow s \simeq t}{(\Gamma', \Gamma \rightarrow s[r]_p \simeq t)\sigma}$$

falls

- $\sigma = mgu(s|_p, l)$
- ◦ $r\sigma, \Gamma'\sigma \not\succeq l\sigma$
- ◦ $t\sigma, \Gamma\sigma \not\succeq s\sigma$
- ◦ $(l \simeq r)\sigma \not\succeq (s \simeq t)\sigma$
- ◦ in Γ, Γ' ist keine Gleichung selektiert

Superposition links:

$$\frac{\Gamma' \rightarrow l \simeq r \qquad \Gamma, \underline{s \simeq t} \rightarrow \Delta}{(\Gamma', \Gamma, s[r]_p \simeq t \rightarrow \Delta)\sigma}$$

falls

- $\sigma = mgu(s|_p, l)$
- ◦ $r\sigma, \Gamma'\sigma \not\succeq l\sigma$
- ◦ $t\sigma \not\succeq s\sigma$
- ◦ in Γ' ist keine Gleichung selektiert

Reflexivitäts-Resolution:

$$\frac{\Gamma, \underline{s \simeq t} \rightarrow \Delta}{(\Gamma \rightarrow \Delta)\sigma}$$

falls

- $\sigma = mgu(s, t)$

Satz 2 Sei S der Abschluss einer Menge S_0 von Hornklauseln bezüglich $\mathcal{H}$. Dann ist S_0 genau dann erfüllbar, wenn S nicht die leere Klausel enthält.

Beweis Das Inferenzsystem $\mathcal{H}$ ist, wie man leicht nachprüfen kann, korrekt. Daher ist S_0 unerfüllbar, wenn $\Box \in S$ gilt. Es gelte also $\Box \notin S$. Um zu zeigen, dass S_0 dann erfüllbar ist, wird ein Modell von S_0 konstruiert.

Sei GS die Menge aller Grundinstanzen der Klauseln aus S. Dann ist GS – wie man leicht nachweisen kann – abgeschlossen bezüglich $\mathcal{G}$ mit Selektion. Mit Hilfe von Satz 1 in der in Abschnitt 2.3 für Selektion angepassten Variante kann man dann ein Reduktionssystem R konstruieren, das Modell für GS ist. Sei $C{\cdot}\sigma$ eine beliebige Grundinstanz einer beliebigen Klausel $C \in S_0 \subseteq S$. Dann gilt $C{\cdot}\sigma \in GS$ und damit $R^* \models C{\cdot}\sigma$. Also erfüllt R alle Grundinstanzen von C. Da R ein Herbrand-Modell ist, ist dies gleichbedeutend mit $R^* \models C$. Also erfüllt R alle Klauseln aus S_0 und ist somit ein Modell für S_0. □

Weiterhin kann man zeigen, dass die Superpositions-Inferenzen nur an Positionen gezogen werden müssen, an denen keine Variablen stehen. Denn greift eine Superpositions-Inferenz im Beweis zu Satz 1 in der in Abschnitt 2.3 für Selektion angepassten Variante das minimale Gegenbeispiel $C{\cdot}\sigma$ an einer solchen Variablenposition an, so liegt die folgende Situation vor: das minimale Gegenbeispiel $C{\cdot}\sigma \in GS$ ist an einer Substitutionsposition p durch eine Regel $l \Rightarrow r \in R$ reduzierbar. Sei σ' die Substitution, die aus σ durch Anwenden der Regel $l \Rightarrow r$ entsteht. Dann ist aber auch die Klausel $C{\cdot}\sigma' \in GS$, da sie ebenfalls eine Grundinstanz von C ist. Außerdem gilt $R^* \not\models C\sigma' \equiv C\sigma[r]_p$, da $R^* \not\models C\sigma$, $C\sigma|_p \equiv l$ und $l \Rightarrow r \in R$. Dies ist ein Widerspruch zur Minimalität von $C{\cdot}\sigma$, da $C{\cdot}\sigma' \prec C{\cdot}\sigma$ wegen $l \succ r$ gilt.

3. Suchraumeinschränkung durch Constraints

3.1. Motivation

Im Folgenden wird der Superpositionskalkül für allgemeine Hornklauseln weiter verfeinert, indem die in Abschnitt 2.4 eingeführten Bedingungen verstärkt werden, unter denen eine Inferenz nicht gezogen werden muss. Wenn eine Inferenz auf Grundklauseln nur dann gezogen werden muss, wenn $s\sigma \succ t\sigma$ gilt, wurde dies dadurch approximiert, dass $t \not\succeq s$ im allgemeinen Fall überprüft wurde. Die Grundinstanzen, für die die Inferenz gezogen werden muss, lassen sich jedoch genauer dadurch einschränken, dass die Konklusion mit dem Constraint $s > t$ versehen wird. Dadurch werden nur noch die Grundinstanzen $C\sigma$ der Konklusion C betrachtet, für die $s\sigma \succ t\sigma$ gilt. Diese Einschränkung gilt auch für weitere Inferenzschritte. Vererbt man die Constraints, so entstehen schnell größere Constraints, die eine noch stärkere Einschränkung der Grundinstanzen bedeuten. Dies führt dazu, dass einige Inferenzen nicht gezogen werden müssen, da die Konklusion keine Grundinstanzen mehr besitzt, das heißt der Constraint unerfüllbar ist. Es ist sinnvoll, neben den Ordnungsbedingungen auch die Unifikationsbedingungen mit in die entstehenden Constraints aufzunehmen und zu vererben. Hierdurch kann man die durch das Anwenden des Unifikators auf die Konklusion entstehenden Substitutionspositionen markieren. Wie im Folgenden bewiesen werden wird, müssen an diesen Substitutionspositionen keine Inferenzen gezogen werden.

Da Constraints nur der Beschränkung des Suchraums dienen, können sie ohne Verlust der Korrektheit oder Vollständigkeit des Inferenzsystems ganz oder teilweise aufgegeben werden. Dabei muss man allerdings eine Vergrößerung des Suchraums in Kauf nehmen. Der Ordnungsanteil eines entstehenden Constraints beschreibt nur Bedingungen, die der Beschränkung des Suchraums dienen, und kann somit ohne Anpassung der Klausel – analog zu den Ordnungsbedingungen in $\mathcal{H}$ – ganz oder teilweise aufgegeben werden. Will man den Ordnungsanteil trotz des Aufgebens zumindest teilweise nutzen, so kann vor dem Aufgeben eines Ordnungsconstraints $s > t$ – analog zu $\mathcal{H}$ – die für seine Erfüllbarkeit notwendige Bedingung $t \not\succeq s$ überprüft werden. Der Gleichheitsanteil eines Constraints ist für die Korrektheit wichtig, darf also nicht ohne Anpassungen aufgegeben werden. Es ist jedoch möglich, einen erfüllbaren Gleichheitsanteil ganz oder teilweise aufzugeben, indem der allgemeinste Unifikator des aufzugebenden Teils bestimmt und auf die Klausel sowie den restlichen Constraint angewendet wird. Ist der Gleichheitsanteil unerfüllbar, so ist der gesamte Constraint unerfüllbar und die Klausel braucht nicht weiter betrachtet zu werden. Auch das Aufgeben von Gleichheitsconstraints führt zu einem Vorgehen wie im Inferenzsystem $\mathcal{H}$. Werden alle Constraints des folgenden Inferenzsystems $\mathcal{H}_{\mathcal{C}}$ aufgegeben, führt dies wieder zu Inferenzsystem $\mathcal{H}$.

3.2. Das Inferenzsystem $\mathcal{H}_\mathcal{C}$

Das folgende Inferenzsystem $\mathcal{H}_\mathcal{C}$ arbeitet auf Hornklauseln mit Constraints. Wie beschrieben wird davon ausgegangen, dass die Prämissen der Inferenzen paarweise variablendisjunkt sind. Außerdem benutzt dieses Inferenzsystem die in Abschnitt 2.3 eingeführte Selektion. Die dort eingeführte Selektionsregel ist zu beachten.

Man beachte, dass die im Inferenzsystem $\mathcal{G}$ für Grundklauseln eingeführten Bedingungen nicht mehr wie im Inferenzsystem $\mathcal{H}$ approximiert werden müssen, sondern sich exakt in den Constraints wiederspiegeln.

Superposition rechts:

$$\frac{\Gamma' \to l \simeq r \mid \aleph \qquad \Gamma \to s \simeq t \mid \beth}{\Gamma', \Gamma \to s[r]_p \simeq t \mid s|_p \doteq l \ \wedge \ l > \Gamma', r \ \wedge \ s > \Gamma, t \ \wedge \ s \simeq t > l \simeq r \ \wedge \ \aleph \ \wedge \ \beth}$$

falls

- der Constraint der Konklusion ist erfüllbar
- $s|_p$ ist keine Variable
- in Γ und Γ' ist keine Gleichung selektiert

Superposition links:

$$\frac{\Gamma' \to l \simeq r \mid \aleph \qquad \Gamma, \underline{s \simeq t} \to \Delta \mid \beth}{\Gamma', \Gamma, s[r]_p \simeq t \to \Delta \mid s|_p \doteq l \ \wedge \ l > \Gamma', r \ \wedge \ s > t \ \wedge \ \aleph \ \wedge \ \beth}$$

falls

- der Constraint der Konklusion ist erfüllbar
- $s|_p$ ist keine Variable
- in Γ' ist keine Gleichung selektiert

Reflexivitäts-Resolution:

$$\frac{\Gamma, \underline{s \simeq t} \to \Delta \mid \aleph}{\Gamma \to \Delta \mid s \doteq t \ \wedge \ \aleph}$$

falls

- der Constraint der Konklusion ist erfüllbar

Bemerkung 1 Durch die Constraintvererbung können schnell sehr große, komplizierte Constraints entstehen, deren Erfüllbarkeit nur schwer zu bestimmen ist. Wie aber oben dargelegt, darf eine Klausel jederzeit durch eine bis auf Variablenumbenennung syntaktisch äquivalente Klausel ersetzt werden. Dies ist erlaubt, da die Menge der Grundinstanzen zweier bis auf Variablenumbenennung syntaktisch äquivalenter Klauseln übereinstimmen. Es ist sehr sinnvoll, dies zu nutzen, um die entstehenden Constraints so zu vereinfachen, dass ihre Erfüllbarkeit leichter zu überprüfen ist. So kann z.B. die Klausel $f(x,a) \simeq b \mid f(x,y) \doteq f(b,y) \ \wedge \ y > b$ zu $f(x,a) \simeq b \mid x \doteq b$ vereinfacht werden. Bei dieser Vereinfachung kann auch Wissen über die verwendete Reduktionsordnung ausgenutzt werden. Näheres ist in Abschnitt 3.3 zu finden.

Bemerkung 2 Der bei der *Superposition rechts* auftretende Constraint $s|_p \doteq l \ \wedge\ l > \Gamma', r \ \wedge\ s > \Gamma, t \ \wedge\ s \simeq t > l \simeq r \ \wedge\ \aleph \ \wedge\ \beth$ wirkt wegen des atomaren Constraints $s \simeq t > l \simeq r$ auf den ersten Blick sehr kompliziert. Dieser atomare Constraint kann aber sehr leicht vereinfacht werden. Gilt $p \neq \lambda$, so wird $s \simeq t > l \simeq r$ von jeder Substitution erfüllt, die den Constraint $s|_p \doteq l \ \wedge\ l > r \ \wedge\ s > t$ erfüllt, und kann somit weggelassen werden. Gilt dagegen $p = \lambda$, so erfüllt eine Substitution, die $s|_p \doteq l$ erfüllt, genau dann $s \simeq t > l \simeq r$, wenn sie den Constraint $t > r$ erfüllt. Daher kann in diesem Fall $s \simeq t > l \simeq r$ zu $t > r$ vereinfacht werden.

Bemerkung 3 Trotz der genannten Vereinfachungen bleiben die entstehenden Constraints meist sehr kompliziert und damit dauert es lange, sie auf Erfüllbarkeit zu überprüfen. Wie aber im Inferenzsystem beschrieben, dient der Test auf Erfüllbarkeit nur der Einschränkung des Suchraums und ist für die Korrektheit unwichtig. Daher kann man wie folgt vorgehen: Falls leicht ersichtlich ist, dass ein Constraint unerfüllbar ist, wird die entsprechende Inferenz nicht gezogen. Ist die Erfüllbarkeit des Constraints dagegen nur schwer feststellbar, so wird die Inferenz gezogen, um die Vollständigkeit nicht zu verlieren, falls der Constraint erfüllbar ist. Bleibt ein Constraint auch dann noch zu kompliziert, kann er – wie oben beschrieben – ganz oder teilweise aufgegeben werden.

Festzulegen bleibt, wann ein Constraint kompliziert ist und in welchen Fällen seine Erfüllbarkeit nur schwer feststellbar ist. Hierzu müssen geeignete Heuristiken entwickelt werden.

Satz 3 Sei S der Abschluss bezüglich $\mathcal{H}_{\mathcal{C}}$ einer Menge S_0 von Hornklauseln ohne Constraint, das heißt von Hornklauseln mit allgemeingültigem Constraint $\top$. Dann ist S_0 genau dann erfüllbar, wenn S nicht die leere Klausel enthält.

Beweis Ist die leere Klausel in S enthalten, so ist S_0 unerfüllbar, da die Inferenzen korrekt sind.

Ist die leere Klausel nicht in S enthalten, ist zu zeigen, dass S_0 erfüllbar ist. Dazu wird ein Modell zu S_0 konstruiert. Dieses wird in Form eines Reduktionssystems R angegeben, das gleichzeitig mit einer Menge GS wie folgt (durch Induktion über die Ordnung auf Klauseln) erzeugt wird:

Eine Grundinstanz $C{\cdot}\sigma$ einer Klausel $C \in S$ gehört genau dann zu GS, wenn $C{\cdot}\sigma$ an allen Substitutionspositionen nicht reduzierbar bezüglich $R_{C\cdot\sigma}$ ist. Da der Constraint $\aleph\sigma$ einer Grundinstanz $(C \mid \aleph){\cdot}\sigma$ nach Definition (vergleiche Abschnitt 1.6) immer erfüllt ist und somit die Menge der Instanzen nicht weiter beeinflussen kann, kann der Constraint für Grundinstanzen und damit alle Klauseln aus GS vernachlässigt werden.

Die Klausel $C{\cdot}\sigma$ generiert die Regel $l\sigma \Rightarrow r\sigma$ gdw. sie die Form $(\Gamma \rightarrow l \simeq r){\cdot}\sigma$ hat und gilt:

1. $C{\cdot}\sigma \in GS$
2. $R^*_{C\cdot\sigma} \not\models C\sigma$
3. $l\sigma \succ r\sigma, \Gamma\sigma$
4. $l\sigma$ ist nicht reduzierbar bezüglich $R_{C\cdot\sigma}$

wobei $R_{C\cdot\sigma}$ die Menge der von allen Klauseln $D\cdot\theta$ aus GS mit $C\cdot\sigma \succ D\cdot\theta$ generierten Regeln ist. R ist die Menge aller von Klauseln aus GS generierten Regeln. $R^{C\cdot\sigma}$ bezeichne die Menge $R_{C\cdot\sigma}$ vereinigt mit der eventuell von $C\cdot\sigma$ generierten Regel.

Man kann sich die Definition von R und GS auch als zweistufigen Prozess vorstellen. Zunächst wird die Menge GS konstruiert (wobei implizit natürlich bereits R erstellt werden muss). GS ist eine Menge von Grundhornklauseln. Definiert man R wie in Satz 1 mit $S := GS$, so entsteht das gleiche R, wie nach der obigen Definition. Diese Überlegungen machen es leicht einsichtlich, dass alle bisherigen Ergebnisse, die nur R betreffen und keine Abgeschlossenheit von R bezüglich eines Inferenzsystems erfordern, übernommen werden dürfen.

Zu jeder Klausel $C_g \in GS$ gibt es eine Klausel $C \mid \aleph \in S$ und eine Substitution σ, so dass $C\sigma \equiv C_g$ gilt und σ den Constraint $\aleph$ erfüllt. Im Folgenden bedeute die Schreibweise $D\cdot\sigma \in GS$ zusätzlich $D \in S$ und σ erfüllt den Constraint der Klausel D. Für zwei Klauseln $C\cdot\sigma$ und $C'\cdot\sigma$ aus GS wird meist die gleiche Substitution σ verwendet. Diese Vereinfachung ist möglich, da C und C' o. B. d. A. als variablendisjunkt angenommen werden dürfen.

Lemma 8 Alle Reduktionsysteme R' mit $R' \subseteq R$ sind konvergent.

Beweis analog Lemma 1

Lemma 9 Sei $C\cdot\sigma$ eine Grundhornklausel. Dann gilt für alle Regeln $l\sigma \Rightarrow r\sigma \in R \setminus R_{C\cdot\sigma}$ und alle Teilterme $s\sigma$ von $C\cdot\sigma$ die Beziehung $l\sigma \succeq s\sigma$. Gilt für einen Teilterm $s\sigma \equiv l\sigma$, so besitzt $C\cdot\sigma$ die Form $(\Gamma \to s \simeq t)\cdot\sigma$ mit $s\sigma \succ t\sigma, \Gamma\sigma$.

Beweis analog Lemma 2

Lemma 10 Sei $C\cdot\sigma \in GS$ eine Grundhornklausel. Dann gilt für alle Regeln $l\sigma \Rightarrow r\sigma \in R \setminus R^{C\cdot\sigma}$ und alle Teilterme $s\sigma$ von $C\cdot\sigma$ die Beziehung $l\sigma \succeq s\sigma$. Gilt für einen Teilterm $s\sigma \equiv l\sigma$, so besitzt $C\cdot\sigma$ die Form $(\Gamma \to s \simeq t)\cdot\sigma$ mit $s\sigma \succ t\sigma, \Gamma\sigma$ und $R^*_{C\cdot\sigma} \not\models \Gamma\sigma$.

Beweis analog Lemma 3

Lemma 11 Sei $C\cdot\sigma := (\Gamma, \Gamma' \to \Delta)\cdot\sigma$ eine Grundhornklausel. Dann gilt für alle Grundhornklauseln $D\cdot\sigma$ mit $D\cdot\sigma \succ C\cdot\sigma$: $R^*_{C\cdot\sigma} \models \Gamma\sigma$ gdw. $R^{C\cdot\sigma^*} \models \Gamma\sigma$ gdw. $R^*_{D\cdot\sigma} \models \Gamma\sigma$ gdw. $R^* \models \Gamma\sigma$.

Beweis analog Lemma 4

Folgerung 3 (Folgerung aus Lemma 11) Für zwei Grundhornklauseln $C\cdot\sigma$ und $D\cdot\sigma$ der Form $C\cdot\sigma := (\Gamma, \Gamma' \to \Delta)\cdot\sigma$ und $D\cdot\sigma := (\Gamma, \Gamma'' \to \Delta')\cdot\sigma$ gilt $R^*_{C\cdot\sigma} \models \Gamma\sigma$ gdw. $R^*_{D\cdot\sigma} \models \Gamma\sigma$.

Lemma 12 Sei $C{\cdot}\sigma$ eine Grundhornklausel. Dann gilt für alle Grundhornklauseln $D{\cdot}\sigma$ mit $D{\cdot}\sigma \succ C{\cdot}\sigma$: gilt $R^* \not\models C{\cdot}\sigma$, so gilt auch $R^*_{C\cdot\sigma} \not\models C{\cdot}\sigma$, $R^{C\cdot\sigma *} \not\models C{\cdot}\sigma$ und $R^*_{D\cdot\sigma} \not\models C{\cdot}\sigma$.

Beweis analog Lemma 5

Lemma 13 Sei $C{\cdot}\sigma \in GS$ eine Grundhornklausel. Dann gilt für alle Grundhornklauseln $D{\cdot}\sigma$ mit $D{\cdot}\sigma \succ C{\cdot}\sigma$: gilt $R^{C\cdot\sigma *} \not\models C{\cdot}\sigma$ so gelten auch $R^*_{D\cdot\sigma} \not\models C{\cdot}\sigma$ und $R^* \not\models C{\cdot}\sigma$.

Beweis analog Lemma 6

Folgerung 4 (Folgerung aus Lemma 12 und Lemma 13) Sei $C{\cdot}\sigma \in GS$ eine Grundhornklausel. Dann gilt für alle Grundhornklauseln $D{\cdot}\sigma$ mit $D{\cdot}\sigma \succ C{\cdot}\sigma$: $R^{C\cdot\sigma *} \models C{\cdot}\sigma$ gdw. $R^*_{D\cdot\sigma} \models C{\cdot}\sigma$ gdw. $R^* \models C{\cdot}\sigma$.

Beweis analog Folgerung 2

Lemma 14 Sei $C{\cdot}\sigma$ eine Grundhornklausel und sei $D{\cdot}\sigma$ eine beliebige Grundhornklausel mit $D{\cdot}\sigma \succ C{\cdot}\sigma$. Weiter sei p eine Position von $C{\cdot}\sigma$, die einen Term bezeichnet, der nicht strikt maximal in $C{\cdot}\sigma$ ist oder in einem negativen Literal vorkommt. Dann sind folgende Aussagen äquivalent:

- $C{\cdot}\sigma$ ist an der Position p reduzierbar bezüglich $R^{C\cdot\sigma}$
- $C{\cdot}\sigma$ ist an der Position p reduzierbar bezüglich $R_{D\cdot\sigma}$
- $C{\cdot}\sigma$ ist an der Position p reduzierbar bezüglich R

Beweis Falls die Klausel $C{\cdot}\sigma$ an der Position p reduzierbar ist bezüglich $R^{C\cdot\sigma}$, ist sie an p auch reduzierbar bezüglich $R_{D\cdot\sigma}$ und R, da $R^{C\cdot\sigma} \subseteq R_{D\cdot\sigma} \subseteq R$.

Sei umgekehrt $C{\cdot}\sigma$ an der Position p nicht reduzierbar bezüglich $R^{C\cdot\sigma}$. Sei $l \Rightarrow r$ eine beliebige Regel aus $R\backslash R^{C\cdot\sigma}$ und weiter sei $u := C\sigma|_p$ der Teilterm von $C{\cdot}\sigma$ an der Position p. Nach Lemma 9 gilt dann $l \succ u$, da u nicht strikt maximal in $C{\cdot}\sigma$ ist oder u in einem negativen Literal auftritt. Also ist $C{\cdot}\sigma$ nicht an der Position p reduzierbar durch die Regel $l \Rightarrow r$. Da $s \Rightarrow t \in R \setminus R^{C\cdot\sigma}$ beliebig und nach Voraussetzung $C{\cdot}\sigma$ an der Position p nicht reduzierbar bezüglich $R^{C\cdot\sigma}$ ist, ist somit $C{\cdot}\sigma$ an der Position p nicht reduzierbar bezüglich R. Also ist auch $C{\cdot}\sigma$ an der Position p nicht reduzierbar bezüglich $R_{D\cdot\sigma} \subseteq R$. $\square$

Lemma 15 Sei $C{\cdot}\sigma := (\Gamma, \underline{u \simeq v} \rightarrow \Delta){\cdot}\sigma$ eine Klausel aus GS mit $u\sigma \equiv v\sigma$. Dann ist $D{\cdot}\sigma := (\Gamma \rightarrow \Delta){\cdot}\sigma \in GS$ und es gilt $D{\cdot}\sigma \prec C{\cdot}\sigma$.

Beweis Die Klausel C besitze die Form $\Gamma, \underline{u \simeq v} \to \Delta \mid \aleph$. Da $C{\cdot}\sigma$ eine Grundinstanz von C ist, erfüllt σ den Constraint $\aleph$. Nach Voraussetzung erfüllt σ außerdem den atomaren Constraint $u \doteq v$. Somit kann auf die Klausel $C \in S$ eine *Reflexivitäts-Resolution* angewendet werden:

$$\frac{\Gamma, \underline{u \simeq v} \to \Delta \mid \aleph}{\Gamma \to \Delta \mid \aleph \;\wedge\; u \doteq v}$$

Sei $D := \Gamma \to \Delta \mid \aleph \;\wedge\; u \doteq v$ die Konklusion dieser Inferenz. Wegen $D{\cdot}\sigma \subset C{\cdot}\sigma$ gilt $D{\cdot}\sigma \equiv (\Gamma \to \Delta){\cdot}\sigma \prec C{\cdot}\sigma$. Da S abgeschlossen bezüglich $\mathcal{H}_\mathcal{C}$ ist, gilt weiter $D \in S$. Dann gilt $D{\cdot}\sigma \in GS$, denn:

- $D{\cdot}\sigma$ ist Grundinstanz von D
- $D{\cdot}\sigma$ ist an Substitutionspositionen nicht reduzierbar bezüglich $R_{D\cdot\sigma}$, da ansonsten $C{\cdot}\sigma$ an Substitutionspositionen reduzierbar bezüglich $R_{C\cdot\sigma} \supseteq R_{D\cdot\sigma}$ wäre. $\square$

Lemma 16 Seien $C{\cdot}\sigma := (\Gamma, \underline{s \simeq t} \to \Delta){\cdot}\sigma$ und $C'{\cdot}\sigma := (\Gamma' \to l \simeq r){\cdot}\sigma$ zwei Klauseln aus GS und p eine Position mit

- $C'{\cdot}\sigma$ ist generierend
- $s\sigma \succ t\sigma$
- in Γ' ist keine Gleichung selektiert
- $s\sigma|_p \equiv l\sigma$

Dann ist $D{\cdot}\sigma := (\Gamma', \Gamma, s[r]_p \simeq t \to \Delta){\cdot}\sigma \in GS$ und es gilt $D{\cdot}\sigma \prec C{\cdot}\sigma$.

Beweis Die Klausel C besitze die Form $C \equiv \Gamma, \underline{s \simeq t} \to \Delta \mid \gimel$ und C' die Form $C' \equiv \Gamma' \to l \simeq r \mid \aleph$. Da $C'{\cdot}\sigma$ generierend ist, gilt $l\sigma \succ r\sigma, \Gamma'\sigma$. Nach Voraussetzung ist $l\sigma$ ein Teilterm von $s\sigma$ und damit gilt $s\sigma \succeq l\sigma$. Da aber $s\sigma$ in einem negativen und $l\sigma$ in einem positiven Literal auftauchen, gilt insgesamt $C{\cdot}\sigma \succ C'{\cdot}\sigma$.

Man kann eine *Superposition links* zwischen C' und C ziehen:

$$\frac{\Gamma' \to l \simeq r \mid \aleph \qquad \Gamma, \underline{s \simeq t} \to \Delta \mid \gimel}{\Gamma', \Gamma, s[r]_p \simeq t \to \Delta \mid s|_p \doteq l \;\wedge\; l \succ \Gamma', r \;\wedge\; s \succ t \;\wedge\; \aleph \;\wedge\; \gimel}$$

da die folgenden Bedingungen gelten:

- der Constraint $s|_p \doteq l \;\wedge\; l \succ \Gamma', r \;\wedge\; s \succ t \;\wedge\; \aleph \;\wedge\; \gimel$ ist erfüllbar, weil er von σ erfüllt wird. Denn es gilt
  - $s\sigma|_p \equiv l\sigma$ nach Voraussetzung
  - $l\sigma \succ \Gamma'\sigma, r\sigma$, weil $C'{\cdot}\sigma$ generierend ist
  - $s\sigma \succ t\sigma$ nach Voraussetzung
  - σ erfüllt die Constraints $\aleph$ und $\gimel$, da $C'{\cdot}\sigma$ und $C{\cdot}\sigma$ Grundinstanzen von C' und C sind.

- $s|_p$ ist keine Variable, denn wäre $s|_p$ eine Variable x, so wäre $x\sigma \equiv s\sigma|_p \equiv l\sigma$. Das ist nicht möglich, da $C{\cdot}\sigma \in GS$ und damit $C{\cdot}\sigma$ nicht reduzierbar bezüglich $R_{C\cdot\sigma}$ an Substitutionspositionen ist. $x\sigma \equiv l\sigma$ ist aber von $l\sigma \Rightarrow r\sigma$ reduzierbar und $l\sigma \Rightarrow r\sigma \in R_{C\cdot\sigma}$, da $C'{\cdot}\sigma \prec C{\cdot}\sigma$ die Regel $l\sigma \Rightarrow r\sigma$ generiert.
- in Γ' ist nach Voraussetzung keine Gleichung selektiert

Die Konklusion dieses Inferenzschrittes heiße D, also $D := \Gamma', \Gamma, s[r]_p \simeq t \rightarrow \Delta \mid s|_p \doteq l \ \wedge \ l \succ \Gamma', r \ \wedge \ s \succ t \ \wedge \ \aleph \ \wedge \ \beth$. Da S abgeschlossen ist bezüglich $\mathcal{H}_\mathcal{C}$, ist die Konklusion $D \in S$. Außerdem gilt $C{\cdot}\sigma \succ D{\cdot}\sigma \equiv (\Gamma', \Gamma, s[r]_p \simeq t \rightarrow \Delta){\cdot}\sigma$ wegen $s\sigma \succ s\sigma[r\sigma]_p$ und $s\sigma \succeq l\sigma \succ \Gamma'\sigma$. Dann ist aber $D{\cdot}\sigma \in GS$, denn:

- $D{\cdot}\sigma$ ist Grundinstanz von D
- $D{\cdot}\sigma$ ist an Substitutionspositionen nicht reduzierbar bezüglich $R_{D\cdot\sigma}$, denn
  - $\Gamma{\cdot}\sigma$, $s{\cdot}\sigma$, $t{\cdot}\sigma$ und $\Delta{\cdot}\sigma$ sind nicht reduzierbar an Substitutionspositionen bezüglich $R_{C\cdot\sigma} \supseteq R_{D\cdot\sigma}$.
  - $\Gamma'{\cdot}\sigma$ und $r{\cdot}\sigma$ sind nicht reduzierbar an Substitutionspositionen bezüglich $R_{C'\cdot\sigma}$ und somit auch nicht bezüglich $R^{C'\cdot\sigma}$, da $R^{C'\cdot\sigma}$ zusätzlich nur die Regel $l\sigma \Rightarrow r\sigma$ enthält und diese nicht anwendbar ist, da $l\sigma \succ \Gamma'\sigma, r\sigma$ gilt (weil $C'{\cdot}\sigma$ generierend ist). Gilt $D{\cdot}\sigma \prec C'{\cdot}\sigma$, so sind $\Gamma'{\cdot}\sigma$ und $r{\cdot}\sigma$ nicht reduzierbar an Substitutionspositionen bezüglich $R_{D\cdot\sigma} \subseteq R_{C\cdot\sigma}$. Gilt dagegen $D{\cdot}\sigma \succ C'{\cdot}\sigma$, so folgt dies aus Lemma 14. □

Lemma 17 Seien $C{\cdot}\sigma := (\Gamma \rightarrow s \simeq t){\cdot}\sigma$ und $C'{\cdot}\sigma := (\Gamma' \rightarrow l \simeq r){\cdot}\sigma$ zwei Klauseln aus GS und p eine Position mit

- $C'{\cdot}\sigma$ ist generierend
- $s\sigma \succ t\sigma, \Gamma\sigma$
- $s\sigma \simeq t\sigma \succ l\sigma \simeq r\sigma$
- in Γ und Γ' ist keine Gleichung selektiert
- $s\sigma|_p \equiv l\sigma$

Dann ist $D{\cdot}\sigma := (\Gamma', \Gamma \rightarrow s[r]_p \simeq t){\cdot}\sigma \in GS$ und es gilt $D{\cdot}\sigma \prec C{\cdot}\sigma$.

Beweis Die Klausel C besitze die Form $C \equiv \Gamma \rightarrow s \simeq t \mid \beth$ und C' die Form $C' \equiv \Gamma' \rightarrow l \simeq r \mid \aleph$. Da $C'{\cdot}\sigma$ generierend ist, ist $l\sigma \simeq r\sigma$ strikt maximal in $C'{\cdot}\sigma$. Nach Voraussetzung ist $s\sigma$ und somit $s\sigma \simeq t\sigma$ strikt maximal in $C{\cdot}\sigma$. Wegen $s\sigma \simeq t\sigma \succ l\sigma \simeq r\sigma$ gilt daher $C{\cdot}\sigma \succ C'{\cdot}\sigma$.

Es kann eine *Superposition rechts* zwischen C' und C gezogen werden:

$$\frac{\Gamma' \rightarrow l \simeq r \mid \aleph \qquad \Gamma \rightarrow s \simeq t \mid \beth}{\Gamma', \Gamma \rightarrow s[r]_p \simeq t \mid s|_p \doteq l \ \wedge \ l > \Gamma', r \ \wedge \ s > \Gamma, t \ \wedge \ s \simeq t > l \simeq r \ \wedge \ \aleph \ \wedge \ \beth}$$

da die folgenden Bedingungen gelten:

- der Constraint $s|_p \doteq l \ \wedge\ l > \Gamma', r \ \wedge\ s > \Gamma, t \ \wedge\ s \simeq t > l \simeq r \ \wedge\ \aleph \ \wedge\ \beth$ ist erfüllbar, weil er von σ erfüllt wird. Denn es gilt
  - $s\sigma|_p \equiv l\sigma$ nach Voraussetzung
  - $l\sigma \succ r\sigma, \Gamma'\sigma$, weil $C'{\cdot}\sigma$ generierend ist
  - $s\sigma \succ t\sigma, \Gamma\sigma$ nach Voraussetzung.
  - $s\sigma \simeq t\sigma \succ l\sigma \simeq r\sigma$ nach Voraussetzung
  - σ erfüllt die Constraints $\aleph$ und $\beth$, da $C'{\cdot}\sigma$ und $C{\cdot}\sigma$ Grundinstanzen von C' und C sind.
- $s|_p$ ist keine Variable, denn wäre $s|_p$ eine Variable x, so wäre $x\sigma \equiv s\sigma|_p \equiv l\sigma$. Das ist nicht möglich, da $C{\cdot}\sigma \in GS$ und damit $C{\cdot}\sigma$ nicht reduzierbar bezüglich $R_{C\cdot\sigma}$ an Substitutionspositionen ist. Der Term $x\sigma \equiv l\sigma$ ist aber von $l\sigma \Rightarrow r\sigma$ reduzierbar und $l\sigma \Rightarrow r\sigma \in R_{C\cdot\sigma}$, da $C'{\cdot}\sigma \prec C{\cdot}\sigma$ die Regel $l\sigma \Rightarrow r\sigma$ generiert.
- in Γ und Γ' ist nach Voraussetzung keine Gleichung selektiert

Die Konklusion dieses Inferenzschrittes heiße D, also $D := \Gamma', \Gamma \rightarrow s[r]_p \simeq t \mid s|_p \doteq l \ \wedge\ l > \Gamma', r \ \wedge\ s > \Gamma, t \ \wedge\ s \simeq t > l \simeq r \ \wedge\ \aleph \ \wedge\ \beth$. Da S abgeschlossen ist bezüglich $\mathcal{H}_{\mathcal{C}}$, ist die Konklusion $D \in S$. Außerdem gilt $C{\cdot}\sigma \succ D{\cdot}\sigma \equiv (\Gamma', \Gamma \rightarrow s[r]_p \simeq t){\cdot}\sigma$ wegen $s\sigma \succ s\sigma[r\sigma]_p$ und $s\sigma \succeq l\sigma \succ \Gamma'\sigma$. Dann ist aber $D{\cdot}\sigma \in GS$, denn:

- $D{\cdot}\sigma$ ist Grundinstanz von D
- $D{\cdot}\sigma$ ist an Substitutionspositionen nicht reduzierbar bezüglich $R_{D\cdot\sigma}$, denn
  - $\Gamma{\cdot}\sigma$, $s{\cdot}\sigma$ und $t{\cdot}\sigma$ sind an Substitutionspositionen nicht reduzierbar bezüglich $R_{C\cdot\sigma} \supseteq R_{D\cdot\sigma}$.
  - $\Gamma'{\cdot}\sigma$ und $r{\cdot}\sigma$ sind nicht reduzierbar an Substitutionspositionen bezüglich $R_{C'\cdot\sigma}$ und somit auch bezüglich $R^{C'\cdot\sigma}$, da $R^{C'\cdot\sigma}$ zusätzlich nur die Regel $l\sigma \Rightarrow r\sigma$ enthält und diese nicht anwendbar ist, da $l\sigma \succ \Gamma'\sigma, r\sigma$ ist (weil $C'{\cdot}\sigma$ generierend ist). Gilt $D{\cdot}\sigma \prec C'{\cdot}\sigma$, so sind $\Gamma'{\cdot}\sigma$ und $r{\cdot}\sigma$ nicht reduzierbar an Substitutionspositionen bezüglich $R_{D\cdot\sigma} \subseteq R_{C\cdot\sigma}$. Gilt dagegen $D{\cdot}\sigma \succ C'{\cdot}\sigma$, so folgt dies aus Lemma 14. □

Lemma 18 Keine Klausel $C{\cdot}\sigma \in GS$, die eine selektierte Gleichung enthält, ist generierend.

Beweis *Dieses Lemma ist Lemma 7 sehr ähnlich, jedoch nicht vollkommen analog, da für den Beweis von Lemma 7 und damit auch für den Beweis dieses Lemmas die Abgeschlossenheit bezüglich des entsprechenden Inferenzsystems benutzt wird. Dank der Lemmata 15 und 16 kann der Beweis trotzdem fast analog erfolgen.*

Angenommen die Behauptung gilt nicht. Dann gibt es eine generierende Klausel $C{\cdot}\sigma \in GS$ mit einer selektierten Gleichung. Sei $C{\cdot}\sigma \equiv (\Gamma, \underline{u \simeq v} \rightarrow s \simeq t){\cdot}\sigma$ die kleinste solche Klausel. Fallunterscheidung nach der Form von $C{\cdot}\sigma$:

- Fall 1: $u\sigma \equiv v\sigma$

  Nach Lemma 15 ist dann $D{\cdot}\sigma := (\Gamma \to s \simeq t){\cdot}\sigma \in GS$. Diese Klausel $D{\cdot}\sigma$ ist generierend, denn:

  1. $D{\cdot}\sigma \in GS$
  2. Es gilt $R^*_{D\cdot\sigma} \not\models D{\cdot}\sigma$, denn:
     Da $C{\cdot}\sigma$ generierend ist, gilt $R^*_{C\cdot\sigma} \not\models C{\cdot}\sigma$ und damit $R^*_{C\cdot\sigma} \models \Gamma\sigma$ und $R^*_{C\cdot\sigma} \not\models s\sigma \simeq t\sigma$. Wegen $R^*_{C\cdot\sigma} \models \Gamma\sigma$ gilt $R^*_{D\cdot\sigma} \models \Gamma\sigma$ nach Folgerung 3. $R^*_{C\cdot\sigma} \not\models s\sigma \simeq t\sigma$ impliziert $R^*_{D\cdot\sigma} \not\models s\sigma \simeq t\sigma$ wegen $R_{D\cdot\sigma} \subseteq R_{C\cdot\sigma}$. Damit gilt $R^*_{D\cdot\sigma} \not\models D{\cdot}\sigma$.
  3. $s\sigma \succ t\sigma, \Gamma\sigma$, da $C{\cdot}\sigma$ generierend
  4. $s\sigma$ ist nicht reduzierbar bezüglich $R_{D\cdot\sigma}$, da $s\sigma$ nicht reduzierbar bezüglich $R_{C\cdot\sigma} \supseteq R_{D\cdot\sigma}$ ist.

  Also ist $D{\cdot}\sigma$ generierend und generiert $s\sigma \Rightarrow t\sigma$. Wegen $C{\cdot}\sigma \succ D{\cdot}\sigma$ gilt damit aber $R^*_{C\cdot\sigma} \models s\sigma \simeq t\sigma$ und somit $R^*_{C\cdot\sigma} \models C{\cdot}\sigma$. Dies ist ein Widerspruch dazu, dass $C{\cdot}\sigma$ generierend ist.

- Fall 2: $u\sigma \not\equiv v\sigma$

  Sei o. B. d. A. $u\sigma \succ v\sigma$. Da $C{\cdot}\sigma$ generierend ist, gilt $R^*_{C\cdot\sigma} \not\models C{\cdot}\sigma$ und somit $u\sigma{\Downarrow_{R_{C\cdot\sigma}}}v\sigma$. Daher gibt es eine Regel $l\sigma \Rightarrow r\sigma \in R_{C\cdot\sigma}$ mit $u\sigma \equiv u\sigma[l\sigma]_p \Rightarrow_{R_{C\cdot\sigma}} u\sigma[r\sigma]_p{\Downarrow_{R_{C\cdot\sigma}}}v\sigma$. Sei $C'{\cdot}\sigma \equiv (\Gamma' \to l \simeq r){\cdot}\sigma \in GS$ die Klausel, die $l\sigma \Rightarrow r\sigma$ generiert. Somit ist $C'{\cdot}\sigma$ eine generierende Klausel mit $C'{\cdot}\sigma \prec C{\cdot}\sigma$. Daher enthält $\Gamma'\sigma$ und damit Γ' kein selektiertes Literal, da $C{\cdot}\sigma$ nach Voraussetzung die kleinste generierende Klausel ist, die ein selektiertes Literal enthält.

  Nach Lemma 16 ist dann auch $D{\cdot}\sigma \equiv (\Gamma', \Gamma, u[r]_p \simeq v \to s \simeq t){\cdot}\sigma \in GS$. Diese Klausel $D{\cdot}\sigma$ ist generierend, denn

  1. $D{\cdot}\sigma \in GS$
  2. $s\sigma$ ist strikt maximal in $D{\cdot}\sigma$, denn
     - $s\sigma \succ t\sigma, v\sigma, \Gamma\sigma$, da $C{\cdot}\sigma$ generierend ist
     - Es gilt $s\sigma \succ u\sigma$, da $C{\cdot}\sigma$ generierend ist. Wegen $l\sigma \succ r\sigma$ gilt weiter $u\sigma \succ (u[r]_p)\sigma$ und somit $s\sigma \succ (u[r]_p)\sigma$.
     - Da $C'{\cdot}\sigma$ generierend ist, gilt $l\sigma \succ \Gamma'\sigma$, woraus zusammen mit $s\sigma \succeq l\sigma$ folgt $s\sigma \succ \Gamma'\sigma$.
  3. Es gilt $R^*_{D\cdot\sigma} \not\models D{\cdot}\sigma$, denn
     - $R^*_{D\cdot\sigma} \models \Gamma\sigma$ gilt wegen $R^*_{C\cdot\sigma} \models \Gamma\sigma$ nach Folgerung 3
     - $R^*_{D\cdot\sigma} \models \Gamma'\sigma$ gilt wegen $R^*_{C'\cdot\sigma} \models \Gamma'\sigma$ nach Folgerung 3
     - Es gilt $R^*_{C\cdot\sigma} \models u\sigma \simeq v\sigma$. Da aber nach Lemma 9 für alle Regeln $l'\sigma \Rightarrow r'\sigma$ aus $R \backslash R_{D\cdot\sigma}$ und damit auch für alle Regeln aus $R_{C\cdot\sigma} \backslash R_{D\cdot\sigma}$ die Beziehung $l'\sigma \succeq s\sigma \succ u\sigma \succ v\sigma$ gilt, muss bereits in $R_{D\cdot\sigma} \subseteq R_{C\cdot\sigma}$ gelten $R^*_{D\cdot\sigma} \models u\sigma \simeq v\sigma$. Weiterhin gilt $l\sigma \Rightarrow r\sigma \in R_{D\cdot\sigma}$, da $C'{\cdot}\sigma \prec D{\cdot}\sigma$. Zusammen mit $R^*_{D\cdot\sigma} \models u\sigma \simeq v\sigma$ folgt $R^*_{D\cdot\sigma} \models (u[r]_p)\sigma \simeq v\sigma$.
     - $R^*_{D\cdot\sigma} \not\models s\sigma \simeq t\sigma$, da nach Voraussetzung $R^*_{C\cdot\sigma} \not\models s\sigma \simeq t\sigma$ und $R_{C\cdot\sigma} \supseteq R_{D\cdot\sigma}$

4. $s\sigma$ ist nicht reduzierbar bezüglich $R_{D\cdot\sigma}$, da $s\sigma$ nicht reduzierbar bezüglich $R_{C\cdot\sigma} \supseteq R_{D\cdot\sigma}$ ist.

Also ist $D\cdot\sigma$ generierend und generiert $s\sigma \Rightarrow t\sigma$. Wegen $C\cdot\sigma \succ D\cdot\sigma$ gilt damit aber $R^*_{C\cdot\sigma} \models s\sigma \simeq t\sigma$ und somit $R^*_{C\cdot\sigma} \models C\cdot\sigma$. Dies ist ein Widerspruch dazu, dass $C\cdot\sigma$ generierend ist. □

Nach dem Beweis der Lemmata kann jetzt der Beweis des Satzes fortgesetzt werden. Dazu wird zunächst gezeigt, dass R ein Modell von GS ist.

Angenommen R wäre kein Modell von GS. Dann existiert eine minimale Klausel $C\cdot\sigma \in GS$ mit $R^* \not\models C\sigma$. Sei $s\sigma$ der maximale Teilterm von $C\sigma$, also gelte für alle Teilterme $u\sigma$ von $C\sigma$ die Beziehung $s\sigma \succeq u\sigma$.
Fallunterscheidung nach dem Vorkommen von $s\sigma$ in $C\sigma$:

- Fall 1: $C\cdot\sigma$ besitzt die Form $(\Gamma \rightarrow s \simeq t)\cdot\sigma$ mit $s\sigma \equiv t\sigma$.

  Dann ist $C\cdot\sigma$ eine Tautologie. Dies ist aber ein Widerspruch zur Voraussetzung $R^* \not\models C\sigma$.

- Fall 2: $C\cdot\sigma$ besitzt die Form $(\Gamma \rightarrow s \simeq t)\cdot\sigma$ mit $s\sigma \not\equiv t\sigma$ und $s\sigma$ kommt nicht in $\Gamma\sigma$ vor. Weiter ist in $\Gamma\sigma$ und damit in Γ nichts selektiert.

  Nach Voraussetzung gelten $C\cdot\sigma \in GS$ und $s\sigma \succ t\sigma, \Gamma\sigma$. Weiter muss wegen $R^* \not\models C\cdot\sigma$ nach Lemma 12 $R^*_{C\cdot\sigma} \not\models C\cdot\sigma$ gelten. Wegen $R^* \not\models C\cdot\sigma$ kann $C\cdot\sigma$ aber nicht generierend sein. Daher muss $s\sigma$ mit einer Regel $l\sigma \Rightarrow r\sigma \in R_{C\cdot\sigma}$ an einer Position p reduzierbar sein. Sei $C'\cdot\sigma := (\Gamma' \rightarrow l \simeq r)\cdot\sigma$ die Klausel, die $l\sigma \Rightarrow r\sigma$ generiert. Somit gilt $C'\cdot\sigma \in GS$ und $C'\cdot\sigma \prec C\cdot\sigma$.

  Dann ist nach Lemma 17 $D\cdot\sigma := (\Gamma', \Gamma \rightarrow s[r]_p \simeq t)\cdot\sigma \in GS$, denn:

  - $C\cdot\sigma$ und $C'\cdot\sigma$ sind zwei Klauseln aus GS
  - $C'\cdot\sigma$ ist nach Definition generierend
  - $s\sigma \succ t\sigma, \Gamma\sigma$ gilt nach Voraussetzung
  - Da $C'\cdot\sigma$ generierend ist, ist $l\sigma \simeq r\sigma \in R_{C'\cdot\sigma} \subseteq R_{C\cdot\sigma}$. Wegen $R^*_{C\cdot\sigma} \not\models C\cdot\sigma$ und damit $R^*_{C\cdot\sigma} \not\models s\sigma \simeq t\sigma$ muss also $s\sigma \simeq t\sigma \not\equiv l\sigma \simeq r\sigma$ gelten. $s\sigma \simeq t\sigma$ und $l\sigma \simeq r\sigma$ sind die strikt maximalen Literale von $C\cdot\sigma$ bzw. $C'\cdot\sigma$ und es gilt $C\cdot\sigma \succ C'\cdot\sigma$. Also muss $s\sigma \simeq t\sigma \succ l\sigma \simeq r\sigma$ gelten.
  - In Γ und Γ' ist keine Gleichung selektiert, da in Γ nach Voraussetzung nichts selektiert ist und in $\Gamma'\sigma$ und damit in Γ' nach Lemma 18 nichts selektiert ist, da $C'\cdot\sigma$ generierend ist.
  - $s\sigma|_p \equiv l\sigma$ nach Wahl von $l\sigma$ und p

  Es gilt $R^* \not\models D\cdot\sigma \equiv (\Gamma', \Gamma \rightarrow s[r]_p \simeq t)\cdot\sigma$, denn

  - Da $C'\cdot\sigma$ generierend ist, gilt $R^*_{C'\cdot\sigma} \not\models C'\cdot\sigma$ und somit $R^* \supseteq R^*_{C'\cdot\sigma} \models \Gamma'\sigma$.
  - $R^* \models \Gamma\sigma$ wegen $R^* \not\models C\cdot\sigma$
  - $R^* \not\models (s[r]_p \simeq t)\sigma$, da wegen $l\sigma \Rightarrow r\sigma \in R$ sonst auch $R^* \models (s[l]_p \simeq t)\sigma$ und damit $R^* \models C\cdot\sigma$ gelten würde.

Weiter gilt $D{\cdot}\sigma \prec C{\cdot}\sigma$. Also ist $D{\cdot}\sigma$ ein kleineres Gegenbeispiel als $C{\cdot}\sigma$. Dies ist aber ein Widerspruch zu Minimalität von $C{\cdot}\sigma$.

- Fall 3: Der maximale Term $s\sigma$ kommt im Antezedenten vor und im Antezedenten ist nichts selektiert.

  Dies ist ein Widerspruch zur Selektionsregel, nach der eine Gleichung im Antezedenten selektiert sein muss, falls der maximale Term $s\sigma$ in diesem vorkommt.

- Fall 4: Im Antezedenten ist eine Gleichung selektiert. Die Klausel $C{\cdot}\sigma$ besitzt also die Form $(\Gamma, \underline{u \simeq v} \to \Delta){\cdot}\sigma$.

  Da $R^* \not\models C{\cdot}\sigma$, gilt $R^* \models u\sigma \simeq v\sigma$.

  - Fall 4.1: Es gilt $u\sigma \equiv v\sigma$. Dann ist nach Lemma 15 die Klausel $D{\cdot}\sigma \equiv (\Gamma \to \Delta){\cdot}\sigma \in GS$. Da $R^* \not\models C{\cdot}\sigma$, gilt $R^* \models \Gamma\sigma$ und $R^* \not\models \Delta\sigma$. Somit $R^* \not\models D{\cdot}\sigma$. Dies ist ein Widerspruch zur Minimalität von $C{\cdot}\sigma$ wegen $D{\cdot}\sigma \prec C{\cdot}\sigma$.
  - Fall 4.2: Es gilt $u\sigma \not\equiv v\sigma$. Sei o. B. d. A. $u\sigma \succ v\sigma$. Da $R^* \models u\sigma \simeq v\sigma$ gilt, muss $u\sigma$ reduzierbar an einer Position p durch eine Regel $l\sigma \Rightarrow r\sigma \in R$ sein. Sei $C'{\cdot}\sigma := (\Gamma' \to l \simeq r){\cdot}\sigma$ die Klausel, die $l\sigma \Rightarrow r\sigma$ generiert. Nach Lemma 18 ist in $\Gamma'\sigma$ und damit in Γ' keine Gleichung selektiert, da $C'{\cdot}\sigma$ generierend ist.

    Somit ist nach Lemma 16 $D{\cdot}\sigma := (\Gamma, \Gamma', u[r]_p \simeq v \to \Delta){\cdot}\sigma \in GS$. Es gilt $R^* \not\models D{\cdot}\sigma$, denn

    - $R^* \models \Gamma\sigma$ wegen $R^* \not\models C{\cdot}\sigma$
    - Da $C'{\cdot}\sigma$ generierend ist, gilt $R^*_{C'\cdot\sigma} \not\models C'{\cdot}\sigma$ und damit $R^*_{C'\cdot\sigma} \models \Gamma'\sigma$. Da weiter $R_{C'\cdot\sigma} \subseteq R$ gilt, folgt $R^* \models \Gamma'\sigma$.
    - $R^* \models (u[r]_p \simeq v)\sigma$, da wegen $l\sigma \Rightarrow r\sigma \in R$ sonst $R^* \not\models (u[l]_p \simeq v)\sigma$ und damit $R^* \models C{\cdot}\sigma$ gelten würde.
    - $R^* \not\models \Delta$ wegen $R^* \not\models C{\cdot}\sigma$

    Wegen $D{\cdot}\sigma \prec C{\cdot}\sigma$ ist dies ein Widerspruch zur Minimalität von $C{\cdot}\sigma$.

Weitere Fälle sind nicht möglich, da $C{\cdot}\sigma$ eine Hornklausel ist und da nach Voraussetzung $\Box \notin S$ und damit $C \not\equiv \Box$ gilt. Da also alle möglichen Fälle zu einem Widerspruch führen, ist gezeigt, dass R ein Modell von GS ist.

Also bleibt zu zeigen, dass R auch ein Modell von S_0 ist. Sei $C{\cdot}\sigma$ eine beliebige Grundinstanz einer beliebigen Klausel aus $S_0 \subseteq S$. Weiter sei die Substitution σ' definiert durch $x\sigma' := x\sigma\Downarrow_R$ für alle $x \in var(C)$. Da $C \in S_0$ und daher keinen Constraint besitzt, ist $C{\cdot}\sigma'$ eine Grundinstanz von $C \in S_0 \subset S$. Nach Definition von σ' ist außerdem $C{\cdot}\sigma'$ nicht reduzierbar an Substitutionspositionen bezüglich R und damit auch nicht bezüglich $R_{C\cdot\sigma'} \subseteq R$. Also gilt $C{\cdot}\sigma' \in GS$ und damit $R^* \models C{\cdot}\sigma'$. Nach Definition von σ' gilt somit $R^* \models C{\cdot}\sigma$. Also erfüllt R alle Grundinstanzen aller Klauseln aus S_0 und ist somit ein Modell für S_0. $\Box$

3.3. Constraintvereinfachung

Wie bereits angedeutet können durch die Constraintvererbung schnell sehr große, komplizierte Constraints entstehen, deren Erfüllbarkeit nur schwer zu bestimmen ist. Es ist daher sinnvoll, wenn immer möglich die auftretenden Constraints zu vereinfachen. Ziel einer solchen Vereinfachung muss es sein, die Erfüllbarkeit eines Constraints möglichst einfach bestimmen und zwei vereinfachte Constraints leicht zu einem neuen vereinfachten Constraint kombinieren zu können.

Wie oben definiert, wird im Zusammenhang mit Unifikation ein reiner Gleichheitsconstraint $\aleph = \{s_1 \doteq t_1 \ \wedge \ \dots \ \wedge \ s_n \doteq t_n\}$ als eine Menge von Termpaaren $\{(s_1, t_1), \dots, (s_n, t_n)\}$ betrachtet. Mit dieser Definition ist dann jede $\aleph$ erfüllende Substitution ein Unifikator von $\aleph$. Da ein allgemeinster Unifikator von $\aleph$ alle Unifikatoren von $\aleph$ und damit alle $\aleph$ erfüllenden Substitutionen beschreibt, beschreibt er auch alle zu $\aleph$ äquivalenten Constraints. Deshalb werden im Folgenden Substitutionen ebenfalls als Constraints betrachtet. Benötigt man einen Repräsentanten der von der Substitution beschriebenen äquivalenten Constraints, so kann für eine Substitution $\{x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n\}$ der Constraint $x_1 \doteq t_1 \ \wedge \ \dots \ \wedge \ x_n \doteq t_n$ verwendet werden. Als eine *Normalform* eines Constraints $\aleph$ wird ein zu $\aleph$ äquivalenter Constraint $\beth$ der Form $mgu(eqpart(\aleph)) \ \wedge \ \beth'$ verstanden, wobei $\beth'$ ein reiner Ordnungsconstraint ist, in dem keine Variable aus $dom(mgu(eqpart(\aleph)))$ auftritt. Für jeden Constraint $\aleph$ ist somit $mgu(eqpart(\aleph)) \ \wedge \ ordpart(\aleph)$ eine Normalform von $\aleph$. Man beachte dabei, dass für jeden erfüllbaren Constraint $\aleph$ der Gleichheitsanteil erfüllbar ist und somit $mgu(eqpart(\aleph))$ existiert. Der Constraint $f(x, y) \doteq f(a, y) \ \wedge \ g(y) \doteq g(b) \ \wedge \ f(y, z) > b$ besitzt z. B. die Normalform $x \doteq a \wedge y \doteq b \wedge f(b, z) > b$. Er besitzt aber auch z. B. die Normalform $x \doteq a \wedge y \doteq b$, da die hier verwendete Reduktionsordnung die Teiltermeigenschaft besitzt und damit alle Substitutionen $f(b, z) > b$ erfüllen.

Man sieht also, dass es sinnvoll ist, nicht nur den Gleichheitsanteil, sondern auch den Ordnungsanteil eines Constraints zu vereinfachen. Leider ist dies jedoch sehr viel komplizierter als das Vereinfachen des Gleichheitsanteils. Man kann gewisse Eigenschaften – wie beispielsweise die oben verwendete Teiltermeigenschaft –, die für alle verwendeten Reduktionsordnungen gelten, zum Vereinfachen verwenden. Neben der Teiltermeigenschaft sind vor allem die Transitivität, die Irreflexivität, die Verträglichkeit mit der Termstruktur und die Grundtotalität der $\succ$-Relation benutzbar.

Als eines von vielen möglichen Vorgehen kann man zum Vereinfachen eines Constraints $\aleph$ zunächst $mgu(eqpart(\aleph))$ und damit $\aleph' := ordpart(\aleph)$ berechnen. Existiert $mgu(eqpart(\aleph))$ nicht, so ist $\aleph$ unerfüllbar. Anschließend kann man versuchen, die Beziehungen zwischen allen in $\aleph'$ vorkommenden Teiltermen aufzubauen. Für dieses Aufbauen kann man die genannten Eigenschaften ausnutzen. Stellt man dabei einen Widerspruch fest, so ist $\aleph$ unerfüllbar. Andernfalls kann man versuchen eine möglichst kleine, einfache Menge $\aleph''$ von diesen hergeleiteten Ordnungsbeziehungen zu finden, die alle anderen hergeleiteten Beziehungen durch die genannten Eigenschaften der Ordnung implizieren. Als vereinfachten Constraint in Normalform erhält man dann $mgu(eqpart(\aleph)) \ \wedge \ \aleph''$.

Man sieht, dass dieses Verfahren sehr aufwändig ist. Trotzdem kann es zwar in vielen, aber nicht in allen Fällen herausfinden, ob der Constraint erfüllbar ist. Dies ist aber auch für die Korrektheit der vorgestellten Inferenzsysteme nicht notwendig, da unter Vergrößerung des Suchraums auch Klauseln mit unerfüllbarem Constraint verwendet werden dürfen. Das Verfahren nutzt – je nach Implementierung mehr oder weniger stark – nur die Eigenschaften grundtotaler Reduktionsordnungen aus. Man kann jedoch eigentlich alle Eigenschaften der verwendeten Reduktionsordnung zum Vereinfachen benutzen und so eine stärkere Vereinfachung erreichen. Es ist auch möglich für spezielle Reduktionsordnungen Verfahren zu entwickeln, die entscheiden, ob ein Constraint erfüllbar ist.

Bisher wurden Constraints nur durch Umformung in äquivalente Constraints vereinfacht. Wie aber oben dargelegt, darf eine Klausel jederzeit durch eine bis auf Variablenumbenennung syntaktisch äquivalente Klausel ersetzt werden. Es ist sehr sinnvoll, auch dies dazu zu nutzen, die Constraints einer Klausel zu vereinfachen. Sei $C \mid \aleph$ eine Klausel mit einem Constraint $\aleph$ in Normalform. Besitzt $\aleph$ die Form $\aleph \equiv x \doteq t \ \wedge\ \beth$ und kommt x nicht in C vor, so ist $C \mid \aleph$ bis auf Variablenumbenennungen syntaktisch äquivalent zu $C \mid \beth$. Gilt $\aleph \equiv x \doteq y \ \wedge\ \beth$, so kann $C \mid \aleph$ zu $C\sigma \mid \beth\sigma$ mit $\sigma := \{x \leftarrow y\}$ vereinfacht werden ohne Informationen über Substitutionspositionen zu verlieren. Ähnliche Vereinfachungen sind auf Klauselebene auch für Ordnungsconstraints möglich, hängen jedoch zum Großteil von der verwendeten Reduktionsordnung ab.

4. Suchraumeinschränkung durch Redundanzelimination

Bisher wurde ausgehend von der Paramodulation die Menge der zu ziehenden Inferenzen und damit die Menge der generierten Klausel durch verschiedene Techniken immer stärker verkleinert. Dabei wurde ausschließlich in Abhängigkeit von den Prämissen und der Konklusion bestimmt, ob eine Inferenz gezogen werden muss. Die anderen Klauseln, die sich bereits in der Klauselmenge befinden, auf der die Inferenzen arbeiten, wurden dabei nicht berücksichtigt. Indem man diese Klauseln berücksichtigt, kann man für weitere Inferenzen zeigen, dass sie nicht gezogen werden müssen. Außerdem ist es damit möglich, für bereits in der Klauselmenge befindliche Klauseln zu zeigen, dass sie unnötig sind und damit entfernt werden können. Solche Klauseln und Inferenzen werden *redundant* genannt. Dieser Begriff der *Redundanz* soll im Folgenden formalisiert werden. Außerdem soll gezeigt werden, dass redundante Inferenzen nicht gezogen werden müssen und redundante Klauseln aus der Klauselmenge entfernt werden dürfen.

Definition 1 Eine Grundklausel $C{\cdot}\sigma$ heißt *redundant bezüglich der Grundklauseln $D_1{\cdot}\theta, \ldots, D_n{\cdot}\theta$ und bezüglich des Reduktionssystems R* genau dann, wenn folgende Bedingungen gelten:

- Gilt $R^* \models D_1{\cdot}\theta, \ldots D_n{\cdot}\theta$, so gilt auch $R^* \models C{\cdot}\sigma$.
- $C{\cdot}\sigma \succ D_i{\cdot}\theta$ für $1 \leq i \leq n$
- Ist $C{\cdot}\sigma$ nicht reduzierbar an Substitutionspositionen bezüglich R, dann sind auch $D_1{\cdot}\theta, \ldots, D_n{\cdot}\theta$ nicht reduzierbar an Substitutionspositionen bezüglich R.

$C{\cdot}\sigma$ heißt *redundant bezüglich einer Klauselmenge S* genau dann, wenn es für jedes grundkonvergente Reduktionssystem R Grundinstanzen $D_1{\cdot}\theta, \ldots, D_n{\cdot}\theta$ von Klauseln $D_1, \ldots, D_n \in S$ gibt, bezüglich derer $C{\cdot}\sigma$ bezüglich R redundant ist. Häufig sind dabei für alle grundkonvergenten Reduktionssysteme die Grundinstanzen identisch, so dass die folgende Definition sinnvoll ist: $C{\cdot}\sigma$ heißt *redundant bezüglich der Grundklauseln $D_1{\cdot}\theta, \ldots, D_n{\cdot}\theta$* genau dann, wenn $C{\cdot}\sigma$ redundant bezüglich $D_1{\cdot}\theta, \ldots, D_n{\cdot}\theta$ und bezüglich jedes grundkonvergenten Reduktionssystems ist. Sind alle Grundinstanzen einer allgemeinen Klausel C bezüglich S redundant, so heißt C selbst *redundant* bezüglich S. Im Folgenden wird meist statt der beiden Substitutionen σ und θ nur eine einzige Substitution verwendet. Dies ist o. B. d. A. möglich, da die Klauseln, aus denen die Grundinstanzen entstehen, als variablendisjunkt angenommen werden dürfen.

Eine Inferenz heißt bezüglich *S redundant*, wenn eine Prämisse oder die Konklusion redundant ist. Eine Klauselmenge S heißt *hergeleitet bis auf Redundanz* aus einer Klauselmenge S_0 durch ein Inferenzsystems $\mathcal{I}$, wenn S durch Ziehen von $\mathcal{I}$-Inferenzen und durch Entfernen redundanter Klauseln aus S_0 entsteht. S heißt *abgeschlossen bis auf Redundanz bezüglich* eines Inferenzsystems $\mathcal{I}$, wenn jede in S ziehbare $\mathcal{I}$-Inferenz bezüglich S redundant ist. Ist eine Klauselmenge S aus einer Klauselmenge S_0 durch ein Inferenzsystem $\mathcal{I}$ bis auf Redundanz hergeleitet und ist S abgeschlossen bezüglich $\mathcal{I}$ bis auf Redundanz, so heißt S *Abschluss bis auf Redundanz von* S_0 *bezüglich* $\mathcal{I}$.

Lemma 19 Die leere Klausel $\Box$ ist bezüglich keiner Klauselmenge S redundant.

Beweis Angenommen $\Box$ ist redundant bezüglich einer Klauselmenge S. Weiter sei R ein beliebiges grundkonvergentes Reduktionssystem. Dann gibt es Grundinstanzen $D_1{\cdot}\sigma, \ldots, D_n{\cdot}\sigma$ von Klauseln $D_1, \ldots, D_n \in S$, bezüglich derer $\Box$ redundant bezüglich R ist. Es muss also für alle $1 \leq i \leq n$ gelten $D_i{\cdot}\sigma \prec \Box$. Da $\Box$ aber die kleinste Klausel bezüglich $\succ_c$ ist, gibt es keine solchen Klauseln und damit gilt $n = 0$. Es gilt aber $R^* \models \emptyset$ und $R^* \not\models \Box$. Also ist $\Box$ nicht bezüglich $D_1{\cdot}\sigma, \ldots, D_n{\cdot}\sigma$ redundant. $\Box$

Lemma 20 Sei S_1 eine Klauselmenge. Die Menge S_2 entstehe aus S_1 durch Entfernen redundanter Klauseln und durch Hinzunahme weiterer Klauseln. Ist eine Klausel redundant bezüglich S_1, dann ist sie auch redundant bezüglich S_2.

Beweis Angenommen die Behauptung gilt nicht. Dann gibt es eine Klausel, die redundant bezüglich S_1, aber nicht bezüglich S_2 ist. Da eine Klausel genau dann redundant ist, wenn alle ihre Grundinstanzen redundant sind, gibt es dann auch eine Grundklausel, die redundant bezüglich S_1, aber nicht bezüglich S_2 ist. Sei $C{\cdot}\sigma$ die kleinste solche Grundklausel. Sei R ein beliebiges grundkonvergentes Reduktionssystem. Dann gibt es also Grundinstanzen $D_1{\cdot}\sigma, \ldots D_n{\cdot}\sigma$ von Klauseln $D_1, \ldots D_n$ aus S_1, bezüglich derer $C{\cdot}\sigma$ redundant bezüglich R ist.

Für jede solche Grundinstanz $D_i{\cdot}\sigma$ gilt: ist $D_i \notin S_2$, so ist D_i redundant bezüglich S_1 und damit ist $D_i{\cdot}\sigma$ redundant bezüglich S_1. Da $D_i{\cdot}\sigma \prec C{\cdot}\sigma$ gilt, ist dann aber $D_i{\cdot}\sigma$ auch redundant bezüglich S_2. Setze $m_i := 1$ und $E_{i,1} := D_i$, falls $D_i \in S_2$. Gilt $D_i \notin S_2$, so seien $E_{i,1}, \ldots, E_{i,m_i}$ die Klauseln aus S_2 bezüglich deren Grundinstanzen $E_{i,1}{\cdot}\sigma, \ldots, E_{i,m_i}{\cdot}\sigma$ die Klausel $D_i{\cdot}\sigma$ redundant bezüglich R ist. Mit dieser Setzung gilt

- Gilt $R^* \models E_{i,1}{\cdot}\sigma, \ldots, E_{i,m_i}{\cdot}\sigma$, so gilt auch $R^* \models D_i{\cdot}\sigma$.
- $C{\cdot}\sigma \succ D_i{\cdot}\sigma \succeq E_{i,j}$ für $1 \leq j \leq m_i$
- Ist $C{\cdot}\sigma$ nicht reduzierbar an Substitutionspositionen bezüglich R, so ist sind auch $D_i{\cdot}\sigma$ und damit $E_{i,1}\sigma{\cdot}, \ldots, E_{i,m_i}{\cdot}\sigma$ nicht reduzierbar an Substitutionspositionen bezüglich R.

Also ist $C{\cdot}\sigma$ redundant bezüglich $E_{1,1}{\cdot}\sigma, \ldots, E_{1,m_1}{\cdot}\sigma, \ldots, E_{n,1}{\cdot}\sigma, \ldots, E_{n,m_n}{\cdot}\sigma$ und bezüglich R. Da R beliebig gewählt war, ist damit $C{\cdot}\sigma$ redundant bezüglich S_2. Dies ist aber ein Widerspruch zur Voraussetzung. $\Box$

Bemerkung 4 Die Grundidee bei der Verwendung von Redundanz ist, dass keine Inferenzen mit redundanten Prämissen gezogen werden müssen, um die leere Klausel herzuleiten. Das Löschen redundanter Klauseln und das Vernachlässigen von Inferenzen mit redundanter Konklusion lassen sich hierauf zurückführen. Dass keine Inferenzen mit redundanter Konklusion gezogen werden müssen, kann man nämlich darauf zurückführen, dass diese Konklusion direkt wieder aus der Klauselmenge entfernt werden dürfte. Und dass redundante Klauseln während des Vervollständigungsprozesses entfernt werden dürfen, ergibt sich daraus, dass die leere Klausel selbst nicht redundant ist und dass redundante Klauseln während der ganzen folgenden Vervollständigung redundant bleiben. Damit werden sie nie mehr für die Herleitung der leeren Klauseln benötigt und können somit vernachlässigt werden, da keine Inferenzen mit redundanten Prämissen gezogen werden müssen. Diese Erkenntnis, dass redundante Klauseln redundant bleiben, ist sehr hilfreich für die Implementierung.

Satz 4 Sei S_0 eine Menge von Hornklauseln ohne Constraint, das heißt eine Menge von Hornklauseln mit allgemeingültigem Constraint $\top$. Weiter sei S_A ein Abschluss bis auf Redundanz der Menge S_0 bezüglich $\mathcal{H_C}$. Dann ist S_0 genau dann erfüllbar, wenn S_A nicht die leere Klausel enthält.

Beweis *Dieser Beweis ist dem Beweis zu Satz 3 sehr ähnlich. Viele Lemmata können ganz oder mit nur minimalen Ergänzungen übernommen werden. Auch die Beweisstruktur ist fast identisch. Es muss nur der Beweis, dass redundante Klauseln für die Herleitung der leeren Klausel unnötig sind, ergänzt werden.*

Ist die leere Klausel in S_A enthalten, so ist S_0 unerfüllbar, da die Inferenzen korrekt sind.

Ist die leere Klausel nicht in S_A enthalten, dann ist zu zeigen, dass S_0 erfüllbar ist. Dazu wird ein Modell zu S_0 konstruiert. Dieses wird in Form eines Reduktionssystems R angegeben, das gleichzeitig mit einer Menge GS wie folgt (durch Induktion über die Ordnung auf Klauseln) erzeugt wird (bis auf die Definition von S analog zu Satz 3):

Sei die Klauselmenge S gegeben über $S := S_A \cup S_0$. Da $\Box \notin S_A$ gilt, S_A Abschluss von S_0 bis auf Redundanz ist und die leere Klausel bezüglich keiner Menge redundant ist, ist $\Box \notin S_0$ und damit $\Box \notin S = S_A \cup S_0$. Außerdem ist S Abschluss von S_0 bis auf Redundanz bezüglich $\mathcal{H_C}$, da S_A ein Abschluss von S_0 bis auf Redundanz bezüglich $\mathcal{H_C}$ ist.

Eine Grundinstanz $C{\cdot}\sigma$ einer Klausel $C \in S$ gehört genau dann zu GS, wenn $C{\cdot}\sigma$ an allen Substitutionspositionen nicht reduzierbar bezüglich $R_{C\cdot\sigma}$ ist. Da der Constraint $\aleph\sigma$ einer Grundinstanz $(C \mid \aleph){\cdot}\sigma$ nach Definition (vergleiche Abschnitt 1.6) immer erfüllt ist und somit die Menge der Instanzen nicht weiter beeinflussen kann, kann der Constraint für Grundinstanzen und damit alle Klauseln aus GS vernachlässigt werden.

Die Klausel $C{\cdot}\sigma$ generiert die Regel $l\sigma \Rightarrow r\sigma$ gdw. sie die Form $(\Gamma \to l \simeq r){\cdot}\sigma$ hat und gilt:

1. $C{\cdot}\sigma \in GS$
2. $R^*_{C\cdot\sigma} \not\models C\sigma$
3. $l\sigma \succ r\sigma, \Gamma\sigma$

4. $l\sigma$ ist nicht reduzierbar bezüglich $R_{C\cdot\sigma}$

wobei $R_{C\cdot\sigma}$ die Menge der von allen Klauseln $D\cdot\theta$ aus GS mit $C\cdot\sigma \succ D\cdot\theta$ generierten Regeln ist. R ist die Menge aller von Klauseln aus GS generierten Regeln. $R^{C\cdot\sigma}$ bezeichne die Menge $R_{C\cdot\sigma}$ vereinigt mit der eventuell von $C\cdot\sigma$ generierten Regel.

Zu jeder Klausel $C_g \in GS$ gibt es eine Klausel $C \mid \aleph \in S$ und eine Substitution σ, so dass $C\sigma \equiv C_g$ gilt und σ den Constraint $\aleph$ erfüllt. Im Folgenden bedeute die Schreibweise $D\cdot\sigma \in GS$ zusätzlich $D \in S$ und σ erfüllt den Constraint der Klausel D. Für zwei Klauseln $C\cdot\sigma$ und $C'\cdot\sigma$ aus GS wird meist die gleiche Substitution σ verwendet. Diese Vereinfachung ist möglich, da C und C' o. B. d. A. als variablendisjunkt angenommen werden dürfen.

R ist ein Modell von GS, wie im Folgenden gezeigt wird. Angenommen R wäre kein Modell von GS. Dann existiert eine minimale Klausel $M\cdot\sigma \in GS$ mit $R^* \not\models M\sigma$. Es kann gezeigt werden, dass keine solche minimale Klausel existieren kann und somit die Annahme falsch ist. Dazu sind aber noch einige Lemmata nötig.

Die Definition von R entspricht der Definition in Satz 3, abgesehen davon, dass S hier nicht abgeschlossen bezüglich $\mathcal{H}_\mathcal{C}$ sein muss, sondern nur die Abgeschlossenheit bis auf Redundanz gegeben ist. Daher können alle bisherigen Ergebnisse, die nur R betreffen und keine Abgeschlossenheit von R bezüglich $\mathcal{H}_\mathcal{C}$ erfordern, übernommen werden. Dies sind insbesondere die Lemmata 8, 9 10, 11, 12, 13 und 14 sowie die Folgerungen 3 und 4.

Auch die Lemmata 15, 16 und 17 können sehr leicht angepasst werden. Jedes dieser Lemmata benutzt als zentralen Beweisschritt das Ziehen einer $\mathcal{H}_\mathcal{C}$-Inferenz und postuliert, dass die Konklusion dieses Inferenzschrittes in S liegt, da nach Voraussetzung die Prämissen in S liegen und im Kontext dieser Lemmata S abgeschlossen unter $\mathcal{H}_\mathcal{C}$ ist. Hier ist nun S nur bis auf Redundanz abgeschlossen unter $\mathcal{H}_\mathcal{C}$. Die Lemmata lassen trotzdem leicht übernehmen, indem zusätzlich vorausgesetzt wird, dass die Prämissen nicht redundant sind, und indem nur bewiesen wird, dass die Konklusion entweder redundant oder in S ist.

Lemma 21 Sei $C\cdot\sigma := (\Gamma, \underline{u \simeq v} \to \Delta)\cdot\sigma$ eine Klausel aus GS mit $u\sigma \equiv v\sigma$. Weiter sei $C\cdot\sigma$ und damit C nicht redundant bezüglich S. Für $D\cdot\sigma := (\Gamma \to \Delta)\cdot\sigma$ gilt dann:

- $D\cdot\sigma$ ist an Substitutionspositionen nicht reduzierbar bezüglich $R_{D\cdot\sigma}$
- $D\cdot\sigma$ ist entweder redundant bezüglich S oder es gilt $D\cdot\sigma \in GS$
- $D\cdot\sigma \prec C\cdot\sigma$

Beweis analog Lemma 15

Lemma 22 Seien $C\cdot\sigma := (\Gamma, \underline{s \simeq t} \to \Delta)\cdot\sigma$ und $C'\cdot\sigma := (\Gamma' \to l \simeq r)\cdot\sigma$ zwei Klauseln aus GS und p eine Position mit

- $C\cdot\sigma$ und $C'\cdot\sigma$ und damit C und C' sind nicht redundant bezüglich S
- $C'\cdot\sigma$ ist generierend

- $s\sigma \succ t\sigma$
- in Γ' ist keine Gleichung selektiert
- $s\sigma|_p \equiv l\sigma$

Für $D{\cdot}\sigma := (\Gamma', \Gamma, s[r]_p \simeq t \to \Delta){\cdot}\sigma$ gilt dann:

- $D{\cdot}\sigma$ ist an Substitutionspositionen nicht reduzierbar bezüglich $R_{D\cdot\sigma}$
- $D{\cdot}\sigma$ ist entweder redundant bezüglich S oder es gilt $D{\cdot}\sigma \in GS$
- $D{\cdot}\sigma \prec C{\cdot}\sigma$

Beweis analog Lemma 16

Lemma 23 Seien $C{\cdot}\sigma := (\Gamma \to s \simeq t){\cdot}\sigma$ und $C'{\cdot}\sigma := (\Gamma' \to l \simeq r){\cdot}\sigma$ zwei Klauseln aus GS und p eine Position mit

- $C{\cdot}\sigma$ und $C'{\cdot}\sigma$ und damit C und C' sind nicht redundant bezüglich S
- $C'{\cdot}\sigma$ ist generierend
- $s\sigma \succ t\sigma, \Gamma\sigma$
- $s\sigma \simeq t\sigma \succ l\sigma \simeq r\sigma$
- in Γ und Γ' ist keine Gleichung selektiert
- $s\sigma|_p \equiv l\sigma$

Für $D{\cdot}\sigma := (\Gamma', \Gamma \to s[r]_p \simeq t){\cdot}\sigma$ gilt dann:

- $D{\cdot}\sigma$ ist an Substitutionspositionen nicht reduzierbar bezüglich $R_{D\cdot\sigma}$
- $D{\cdot}\sigma$ ist entweder redundant bezüglich S oder es gilt $D{\cdot}\sigma \in GS$
- $D{\cdot}\sigma \prec C{\cdot}\sigma$

Beweis analog Lemma 17

Lemma 24 Sei $M{\cdot}\sigma$ das minimale Gegenbeispiel. Keine Grundklausel $C{\cdot}\sigma$ mit $C{\cdot}\sigma \preceq M{\cdot}\sigma$, die an Substitutionspositionen nicht reduzierbar bezüglich $R_{C\cdot\sigma}$ ist und für die $R^*_{C\cdot\sigma} \not\models C{\cdot}\sigma$ gilt, ist redundant bezüglich S.

Beweis Angenommen $C{\cdot}\sigma$ ist redundant bezüglich S. Da $R_{C\cdot\sigma}$ nach Lemma 8 ein grundkonvergentes Reduktionssystem ist, gibt es Grundinstanzen $D_1{\cdot}\sigma, \ldots, D_n{\cdot}\sigma$ von Klauseln aus S, bezüglich derer $C{\cdot}\sigma$ redundant bezüglich $R_{C\cdot\sigma}$ ist.

Nach Voraussetzung ist $C{\cdot}\sigma$ nicht reduzierbar an Substitutionspositionen bezüglich $R_{C\cdot\sigma}$. Somit ist für alle $1 \leq i \leq n$ auch $D_i{\cdot}\sigma$ nicht reduzierbar an Substitutionspositionen bezüglich $R_{C\cdot\sigma}$ und daher auch nicht reduzierbar bezüglich $R_{D_i\cdot\sigma} \subseteq R_{C\cdot\sigma}$. Also gilt $D_i{\cdot}\sigma \in GS$ für alle $1 \leq i \leq n$. Damit gilt wegen $D_i{\cdot}\sigma \prec C{\cdot}\sigma \preceq M{\cdot}\sigma$ weiter $R^* \models D_i{\cdot}\sigma$ für alle $1 \leq i \leq n$. Nach Folgerung 4 gilt dann aber auch $R^*_{C\cdot\sigma} \models D_i{\cdot}\sigma$ für alle $1 \leq i \leq n$ und somit $R^*_{C\cdot\sigma} \models C{\cdot}\sigma$. Dies ist aber ein Widerspruch zur Voraussetzung. □

Folgerung 5 (Folgerung aus Lemma 24) Sei $M{\cdot}\sigma$ das minimale Gegenbeispiel. Keine generierende Klausel $C{\cdot}\sigma$ mit $C{\cdot}\sigma \prec M{\cdot}\sigma$ ist redundant bezüglich S.

Folgerung 6 (Folgerung aus Lemma 24 und Lemma 12) Sei $M{\cdot}\sigma$ das minimale Gegenbeispiel. Keine Grundhornklausel $C{\cdot}\sigma$ mit $C{\cdot}\sigma \preceq M{\cdot}\sigma$, die an Substitutionspositionen nicht reduzierbar bezüglich $R_{C\cdot\sigma}$ ist und für die $R^* \not\models C{\cdot}\sigma$ gilt, ist redundant bezüglich S.

Folgerung 7 (Folgerung aus Folgerung 6) Das minimale Gegenbeispiel $M{\cdot}\sigma$ ist nicht redundant bezüglich S.

Lemma 25 Keine Klausel $C{\cdot}\sigma \in GS$ mit $C{\cdot}\sigma \prec M{\cdot}\sigma$, die eine selektierte Gleichung enthält, ist generierend.

Beweis *Dieses Lemma ist Lemma 18 sehr ähnlich, jedoch nicht vollkommen analog, da hier zusätzlich gezeigt werden muss, dass weder $C{\cdot}\sigma$ noch die im Beweis hergeleiteten Klauseln, die den Widerspruch verursachen, redundant bezüglich S sind.*

Angenommen die Behauptung gilt nicht. Dann gibt es eine generierende Klausel $C{\cdot}\sigma \prec M{\cdot}\sigma$ in GS mit einer selektierten Gleichung. Sei $C{\cdot}\sigma \equiv (\Gamma, \underline{u \simeq v} \rightarrow s \simeq t){\cdot}\sigma$ die kleinste solche Klausel. Nach Folgerung 5 gilt daher, dass $C{\cdot}\sigma$ nicht redundant bezüglich S ist. Fallunterscheidung nach der Form von $C{\cdot}\sigma$:

- Fall 1: $u\sigma \equiv v\sigma$

  Sei $D{\cdot}\sigma := (\Gamma \rightarrow s \simeq t){\cdot}\sigma$. Nach Lemma 24 ist $D{\cdot}\sigma$ nicht redundant in S, denn:

  - $D{\cdot}\sigma$ ist nach Lemma 21 nicht reduzierbar an Substitutionspositionen bezüglich $R_{D\cdot\sigma}$.
  - Ebenfalls nach Lemma 21 gilt $D{\cdot}\sigma \prec C{\cdot}\sigma \prec M{\cdot}\sigma$
  - Es gilt $R^*_{D\cdot\sigma} \not\models D{\cdot}\sigma$, denn:
    Da $C{\cdot}\sigma$ generierend ist, gilt $R^*_{C\cdot\sigma} \not\models C{\cdot}\sigma$ und damit $R^*_{C\cdot\sigma} \models \Gamma\sigma$ und $R^*_{C\cdot\sigma} \not\models s\sigma \simeq t\sigma$. Wegen $R^*_{C\cdot\sigma} \models \Gamma\sigma$ gilt $R^*_{D\cdot\sigma} \models \Gamma\sigma$ nach Folgerung 3. $R^*_{C\cdot\sigma} \not\models s\sigma \simeq t\sigma$ impliziert $R^*_{D\cdot\sigma} \not\models s\sigma \simeq t\sigma$ wegen $R_{D\cdot\sigma} \subseteq R_{C\cdot\sigma}$. Damit gilt $R^*_{D\cdot\sigma} \not\models D{\cdot}\sigma$.

  Nach Lemma 21 ist also $D{\cdot}\sigma \equiv (\Gamma \rightarrow s \simeq t){\cdot}\sigma \in GS$. Diese Klausel $D{\cdot}\sigma$ ist generierend, denn:

  1. $D{\cdot}\sigma \in GS$
  2. $R^*_{D\cdot\sigma} \not\models D{\cdot}\sigma$
  3. $s\sigma \succ t\sigma, \Gamma\sigma$, da $C{\cdot}\sigma$ generierend ist
  4. $s\sigma$ ist nicht reduzierbar bezüglich $R_{D\cdot\sigma}$, da $s\sigma$ nicht reduzierbar bezüglich $R_{C\cdot\sigma} \supseteq R_{D\cdot\sigma}$ ist.

  Also ist $D{\cdot}\sigma$ generierend und generiert $s\sigma \Rightarrow t\sigma$. Wegen $C{\cdot}\sigma \succ D{\cdot}\sigma$ gilt damit aber $R^*_{C\cdot\sigma} \models s\sigma \simeq t\sigma$ und somit $R^*_{C\cdot\sigma} \models C{\cdot}\sigma$. Dies ist ein Widerspruch dazu, dass $C{\cdot}\sigma$ generierend ist.

- Fall 2: $u\sigma \not\equiv v\sigma$

  Sei o. B. d. A. $u\sigma \succ v\sigma$. Da $C{\cdot}\sigma$ generierend ist, gilt $R^*_{C\cdot\sigma} \not\models C{\cdot}\sigma$ und somit $u\sigma{\Downarrow_{R_{C\cdot\sigma}}}v\sigma$. Daher gibt es eine Regel $l\sigma \Rightarrow r\sigma \in R_{C\cdot\sigma}$ mit $u\sigma \equiv u\sigma[l\sigma]_p \Rightarrow_{R_{C\cdot\sigma}} u\sigma[r\sigma]_p{\Downarrow_{R_{C\cdot\sigma}}}v\sigma$. Sei $C'{\cdot}\sigma := (\Gamma' \rightarrow l \simeq r){\cdot}\sigma \in GS$ die Klausel, die $l\sigma \Rightarrow r\sigma$ generiert. Somit ist $C'{\cdot}\sigma$ eine generierende Klausel mit $C'{\cdot}\sigma \prec C{\cdot}\sigma \prec M{\cdot}\sigma$. Daher enthält $\Gamma'\sigma$ und damit Γ' kein selektiertes Literal, da $C{\cdot}\sigma$ nach Voraussetzung die kleinste generierende Klausel ist, die ein selektiertes Literal enthält. Nach Folgerung 5 ist $C'{\cdot}\sigma$ außerdem nicht redundant bezüglich S.

  Für $D{\cdot}\sigma := (\Gamma', \Gamma, u[r]_p \simeq v \rightarrow s \simeq t){\cdot}\sigma$ gilt:

  - $s\sigma$ ist strikt maximal in $D{\cdot}\sigma$, denn
    * $s\sigma \succ t\sigma, v\sigma, \Gamma\sigma$, da $C{\cdot}\sigma$ generierend ist
    * Es gilt $s\sigma \succ u\sigma$, da $C{\cdot}\sigma$ generierend ist. Wegen $l\sigma \succ r\sigma$ gilt weiter $u\sigma \succ (u[r]_p)\sigma$ und somit $s\sigma \succ (u[r]_p)\sigma$.
    * Da $C'{\cdot}\sigma$ generierend ist, gilt $l\sigma \succ \Gamma'\sigma$, woraus zusammen mit $s\sigma \succeq l\sigma$ folgt $s\sigma \succ \Gamma'\sigma$.
  - Es gilt $R^*_{D\cdot\sigma} \not\models D{\cdot}\sigma$, denn
    * $R^*_{D\cdot\sigma} \models \Gamma\sigma$ gilt wegen $R^*_{C\cdot\sigma} \models \Gamma\sigma$ nach Folgerung 3
    * $R^*_{D\cdot\sigma} \models \Gamma'\sigma$ gilt wegen $R^*_{C'\cdot\sigma} \models \Gamma'\sigma$ nach Folgerung 3
    * Es gilt $R^*_{C\cdot\sigma} \models u\sigma \simeq v\sigma$. Da aber nach Lemma 9 für alle Regeln $l'\sigma \Rightarrow r'\sigma$ aus $R \backslash R_{D\cdot\sigma}$ und damit auch für alle Regeln aus $R_{C\cdot\sigma} \backslash R_{D\cdot\sigma}$ die Beziehung $l'\sigma \succeq s\sigma \succ u\sigma \succ v\sigma$ gilt, muss bereits in $R_{D\cdot\sigma} \subseteq R_{C\cdot\sigma}$ gelten $R^*_{D\cdot\sigma} \models u\sigma \simeq v\sigma$. Weiterhin gilt $l\sigma \Rightarrow r\sigma \in R_{D\cdot\sigma}$, da $C'{\cdot}\sigma \prec D{\cdot}\sigma$. Zusammen mit $R^*_{D\cdot\sigma} \models u\sigma \simeq v\sigma$ folgt $R^*_{D\cdot\sigma} \models (u[r]_p)\sigma \simeq v\sigma$.
    * $R^*_{D\cdot\sigma} \not\models s\sigma \simeq t\sigma$, da nach Voraussetzung $R^*_{C\cdot\sigma} \not\models s\sigma \simeq t\sigma$ und $R_{C\cdot\sigma} \supseteq R_{D\cdot\sigma}$

  Nach Lemma 24 ist $D{\cdot}\sigma$ somit nicht redundant in S, denn:

  - $R^*_{D\cdot\sigma} \not\models D{\cdot}\sigma$
  - $D{\cdot}\sigma$ ist nach Lemma 22 nicht reduzierbar an Substitutionspositionen bezüglich $R_{D\cdot\sigma}$.
  - Ebenfalls nach Lemma 22 gilt $D{\cdot}\sigma \prec C{\cdot}\sigma \prec M{\cdot}\sigma$

  Nach Lemma 22 ist also $D{\cdot}\sigma \equiv (\Gamma', \Gamma, u[r]_p \simeq v \rightarrow s \simeq t){\cdot}\sigma \in GS$. Diese Klausel $D{\cdot}\sigma$ ist generierend, denn:

  1. $D{\cdot}\sigma \in GS$
  2. $s\sigma$ ist strikt maximal in $D{\cdot}\sigma$
  3. $R^*_{D\cdot\sigma} \not\models D{\cdot}\sigma$
  4. $s\sigma$ ist nicht reduzierbar bezüglich $R_{D\cdot\sigma}$, da $s\sigma$ nicht reduzierbar bezüglich $R_{C\cdot\sigma} \supseteq R_{D\cdot\sigma}$ ist.

  Also ist $D{\cdot}\sigma$ generierend und generiert $s\sigma \Rightarrow t\sigma$. Wegen $C{\cdot}\sigma \succ D{\cdot}\sigma$ gilt damit aber $R^*_{C\cdot\sigma} \models s\sigma \simeq t\sigma$ und somit $R^*_{C\cdot\sigma} \models C{\cdot}\sigma$. Dies ist ein Widerspruch dazu, dass $C{\cdot}\sigma$ generierend ist. □

Nach dem Beweis der Lemmata kann jetzt der Beweis des Satzes fortgesetzt werden. Sei $s\sigma$ der maximale Teilterm des minimalen Gegenbeispiels $M{\cdot}\sigma$, also gelte für alle Teilterme $u\sigma$ von $M\sigma$ die Beziehung $s\sigma \succeq u\sigma$.

Fallunterscheidung nach dem Vorkommen von $s\sigma$ in $M{\cdot}\sigma$:

- Fall 1: $M{\cdot}\sigma$ besitzt die Form $(\Gamma \to s \simeq t){\cdot}\sigma$ mit $s\sigma \equiv t\sigma$.

  Dann ist $M{\cdot}\sigma$ eine Tautologie, was ein Widerspruch zur Voraussetzung $R^* \not\models M{\cdot}\sigma$ ist.

- Fall 2: $M{\cdot}\sigma$ besitzt die Form $(\Gamma \to s \simeq t){\cdot}\sigma$ mit $s\sigma \not\equiv t\sigma$ und $s\sigma$ kommt nicht in $\Gamma\sigma$ vor. Weiter ist in $\Gamma\sigma$ und damit in Γ nichts selektiert.

  Nach Voraussetzung gelten $M{\cdot}\sigma \in GS$ und $s\sigma \succ t\sigma, \Gamma\sigma$. Weiter muss wegen $R^* \not\models M{\cdot}\sigma$ nach Lemma 12 $R^*_{M\cdot\sigma} \not\models M{\cdot}\sigma$ gelten. Wegen $R^* \not\models M{\cdot}\sigma$ kann $M{\cdot}\sigma$ aber nicht generierend sein. Daher muss $s\sigma$ mit einer Regel $l\sigma \Rightarrow r\sigma \in R_{M\cdot\sigma}$ an einer Position p reduzierbar sein. Sei $C'{\cdot}\sigma := (\Gamma' \to l \simeq r){\cdot}\sigma$ die Klausel, die $l\sigma \Rightarrow r\sigma$ generiert. Somit gilt $C'{\cdot}\sigma \in GS$ und $C'{\cdot}\sigma \prec M{\cdot}\sigma$.

  Dann ist nach Lemma 23 $D{\cdot}\sigma := (\Gamma', \Gamma \to s[r]_p \simeq t){\cdot}\sigma$ nicht reduzierbar an Substitutionspositionen bezüglich $R_{D\cdot\sigma}$ und $D{\cdot}\sigma$ ist entweder redundant bezüglich S oder es gilt $D{\cdot}\sigma \in GS$, denn:

  - $M{\cdot}\sigma$ und $C'{\cdot}\sigma$ sind zwei Klauseln aus GS
  - $M{\cdot}\sigma$ ist nach Folgerung 7 nicht redundant bezüglich S
  - $C'{\cdot}\sigma$ ist nach Folgerung 5 nicht redundant bezüglich S
  - $C'{\cdot}\sigma$ ist nach Definition generierend
  - $s\sigma \succ t\sigma, \Gamma\sigma$ gilt nach Voraussetzung
  - Da $C'{\cdot}\sigma$ generierend ist, ist $l\sigma \simeq r\sigma \in R_{C'\cdot\sigma} \subseteq R_{C\cdot\sigma}$. Wegen $R^*_{C\cdot\sigma} \not\models C{\cdot}\sigma$ und damit $R^*_{C\cdot\sigma} \not\models s\sigma \simeq t\sigma$ muss also $s\sigma \simeq t\sigma \not\equiv l\sigma \simeq r\sigma$ gelten. $s\sigma \simeq t\sigma$ und $l\sigma \simeq r\sigma$ sind die strikt maximalen Literale von $C{\cdot}\sigma$ bzw. $C'{\cdot}\sigma$ und es gilt $C{\cdot}\sigma \succ C'{\cdot}\sigma$. Also muss $s\sigma \simeq t\sigma \succ l\sigma \simeq r\sigma$ gelten.
  - in Γ ist nach Voraussetzung nichts selektiert
  - $s\sigma|_p \equiv l\sigma$ nach Wahl von $l\sigma$ und p

  Es gilt $R^* \not\models D{\cdot}\sigma \equiv (\Gamma', \Gamma \to s[r]_p \simeq t){\cdot}\sigma$, denn

  - Da $C'{\cdot}\sigma$ generierend ist, gilt $R^*_{C'\cdot\sigma} \not\models C'{\cdot}\sigma$ und somit $R^* \supseteq R^*_{C'\cdot\sigma} \models \Gamma'\sigma$.
  - $R^* \models \Gamma\sigma$ wegen $R^* \not\models C{\cdot}\sigma$
  - $R^* \not\models (s[r]_p \simeq t)\sigma$, da wegen $l\sigma \Rightarrow r\sigma \in R$ sonst auch $R^* \models (s[l]_p \simeq t)\sigma$ und damit $R^* \models C{\cdot}\sigma$ gelten würde.

  Weiter gilt $D{\cdot}\sigma \prec M{\cdot}\sigma$. Nach Folgerung 6 ist also $D{\cdot}\sigma$ nicht redundant bezüglich S. Daher muss $D{\cdot}\sigma \in GS$ gelten. Also ist $D{\cdot}\sigma$ ein kleineres Gegenbeispiel als $M{\cdot}\sigma$. Dies ist aber ein Widerspruch zur Minimalität von $M{\cdot}\sigma$.

- Fall 3: Der maximale Term $s\sigma$ kommt im Antezedenten vor und im Antezedenten ist nichts selektiert.

  Dies ist ein Widerspruch zur Selektionsregel, nach der eine Gleichung im Antezedenten selektiert sein muss, falls der maximale Term $s\sigma$ in diesem vorkommt.

- Fall 4: Im Antezedenten ist eine Gleichung selektiert. Die Klausel $M{\cdot}\sigma$ besitzt also die Form $(\Gamma, \underline{u \simeq v} \rightarrow \Delta){\cdot}\sigma$.

  Da $R^* \not\models M{\cdot}\sigma$, gilt $R^* \models u\sigma \simeq v\sigma$.

  - Fall 4.1: Es gilt $u\sigma \equiv v\sigma$.

    Dann ist nach Lemma 21 die Klausel $D{\cdot}\sigma := (\Gamma \rightarrow \Delta){\cdot}\sigma$ nicht reduzierbar an Substitutionspositionen bezüglich $R_{D\cdot\sigma}$. Außerdem ist $D{\cdot}\sigma$ entweder redundant bezüglich S oder es gilt $D{\cdot}\sigma \in GS$. Da $R^* \not\models M{\cdot}\sigma$, gilt $R^* \models \Gamma\sigma$ und $R^* \not\models \Delta\sigma$. Somit $R^* \not\models D{\cdot}\sigma$. Zusammen mit $D{\cdot}\sigma \prec M{\cdot}\sigma$ folgt aus Folgerung 6, dass $D{\cdot}\sigma$ nicht redundant bezüglich S ist und dass somit $D{\cdot}\sigma \in GS$ gelten muss. Dies ist ein Widerspruch zur Minimalität von $M{\cdot}\sigma$ wegen $D{\cdot}\sigma \prec M{\cdot}\sigma$.
  - Fall 4.2: Es gilt $u\sigma \not\equiv v\sigma$.

    Sei o. B. d. A. $u\sigma \succ v\sigma$. Da $R^* \models u\sigma \simeq v\sigma$ gilt, muss $u\sigma$ reduzierbar an einer Position p durch eine Regel $l\sigma \Rightarrow r\sigma \in R$ sein. Sei $C'{\cdot}\sigma := (\Gamma' \rightarrow l \simeq r){\cdot}\sigma$ die Klausel, die $l\sigma \Rightarrow r\sigma$ generiert. Nach Lemma 25 ist in $\Gamma'\sigma$ und damit in Γ' keine Gleichung selektiert, da $C'{\cdot}\sigma \prec M{\cdot}\sigma$ generierend ist. $M{\cdot}\sigma$ ist nach Folgerung 7 und $C'{\cdot}\sigma$ nach Folgerung 5 nicht redundant bezüglich S.

    Somit ist nach Lemma 22 $D{\cdot}\sigma := (\Gamma, \Gamma', u[r]_p \simeq v \rightarrow \Delta){\cdot}\sigma$ nicht reduzierbar an Substitutionspositionen bezüglich $R_{D\cdot\sigma}$. Außerdem ist $D{\cdot}\sigma$ entweder redundant bezüglich S oder es gilt $D{\cdot}\sigma \in GS$. Es gilt $R^* \not\models D{\cdot}\sigma$, denn

    * $R^* \models \Gamma\sigma$ wegen $R^* \not\models M{\cdot}\sigma$
    * Da $C'{\cdot}\sigma$ generierend ist, gilt $R^*_{C'\cdot\sigma} \not\models C'{\cdot}\sigma$ und damit $R^*_{C'\cdot\sigma} \models \Gamma'\sigma$. Da weiter $R_{C'\cdot\sigma} \subseteq R$ gilt, folgt $R^* \models \Gamma'\sigma$.
    * $R^* \models (u[r]_p \simeq v)\sigma$, da wegen $l\sigma \Rightarrow r\sigma \in R$ sonst $R^* \not\models (u[l]_p \simeq v)\sigma$ und damit $R^* \models M{\cdot}\sigma$ gelten würde.
    * $R^* \not\models \Gamma$ wegen $R^* \not\models M{\cdot}\sigma$

    Zusammen mit $D{\cdot}\sigma \prec M{\cdot}\sigma$ folgt aus Folgerung 6, dass $D{\cdot}\sigma$ nicht redundant bezüglich S ist und dass somit $D{\cdot}\sigma \in GS$ gelten muss. Dies ist ein Widerspruch zur Minimalität von $M{\cdot}\sigma$ wegen $R^* \not\models D{\cdot}\sigma$ und $D{\cdot}\sigma \prec M{\cdot}\sigma$.

Weitere Fälle sind nicht möglich, da $C{\cdot}\sigma$ eine Grundhornklausel ist und da nach Voraussetzung $\Box \notin S$ und somit $C \not\equiv \Box$ gilt. Da also alle möglichen Fälle zu einem Widerspruch führen, ist gezeigt, dass R ein Modell von GS ist.

Also bleibt zu zeigen, dass R auch ein Modell von S_0 ist. Sei $C{\cdot}\sigma$ eine beliebige Grundinstanz einer beliebigen Klausel aus $S_0 \subseteq S$. Weiter sei die Substitution σ' definiert durch $x\sigma' := x\sigma{\Downarrow_R}$ für alle $x \in \mathit{var}(C)$. Da $C \in S_0$ und daher keinen Constraint besitzt, ist $C{\cdot}\sigma'$ eine Grundinstanz von $C \in S_0 \subseteq S$. Nach Definition von σ' ist außerdem $C{\cdot}\sigma'$ nicht reduzierbar an Substitutionspositionen bezüglich R und damit auch nicht bezüglich $R_{C\cdot\sigma'} \subseteq R$. Also gilt $C{\cdot}\sigma' \in GS$ und damit $R^* \models C{\cdot}\sigma'$. Nach Definition von σ' gilt somit $R^* \models C{\cdot}\sigma$. Also erfüllt R alle Grundinstanzen aller Klauseln aus S_0 und ist somit ein Modell für S_0. $\Box$

5. Praktische Realisierung

Um herauszufinden, welche Vor- und Nachteile die Verwendung von Constraints in der Praxis bringt und wie man Constraints am sinnvollsten einsetzen kann, müssen die zuvor beschriebenen theoretischen Überlegungen in der Praxis erprobt werden. Beispielsweise müssen Heuristiken für die Aufgabe von Constraints untersucht werden. Denn einerseits schränken starke Constraints den Suchraum ein, andererseits sind der Test, ob sie erfüllbar sind, und ihre Verwaltung sehr aufwändig. Außerdem können Constraints dazu führen, dass statt einer Klausel $C \mid \aleph$, die viele Grundinstanzen abdeckt, mehrere Klauseln $D_i \mid \beth_i$ hergeleitet werden, die jeweils nur einen Teil der Grundinstanzen abdecken. Obwohl vielleicht die Menge der Grundinstanzen der Klauseln $D_i \mid \beth_i$ eine echte Teilmenge der Menge der Grundinstanzen der Klausel $C \mid \aleph$ ist und somit eine Beschränkung des Suchraums stattgefunden hat, bedeutet dies in der Praxis erheblich mehr Aufwand, da nun der Suchraum nicht mehr so kompakt dargestellt und durchsucht werden kann. Constraints beeinflussen auch die Redundanz von Klauseln. Eine Heuristik zu entwickeln, ist hierbei besonders trickreich, da eine Klausel umso eher redundant ist, je stärker ihr Constraint und je schwächer die Constraints der anderen Klauseln sind.

Man sieht also, dass es trotz der durch die Constraints erreichten Suchraumeinschränkung keineswegs klar ist, ob die Verwendung von Constraints in der Praxis sinnvoll ist und zu einer Leistungssteigerung eines automatischen Theorembeweisers führt.

Eine weitere interessante Frage ist, wie man am geschicktesten mit Redundanz umgeht. Es ist unentscheidbar, ob eine Klausel redundant ist, da sich das Erfüllbarkeitsproblem auf diese Fragestellung reduzieren lässt. Es ist aber auch nicht nötig, für jede Klausel zu entscheiden, ob sie redundant ist, da redundante Klausel zwar aus der Klauselmenge entfernt werden können, aber nicht müssen und ebenso redundante Inferenzen gezogen werden dürfen. Es ist sogar so, dass das Ziehen redundanter Inferenzen das Finden des Beweises beschleunigen kann. Es müssen also *hinreichende Redundanzkriterien* entwickelt werden, die effizient zu überprüfen sind.

Der eingeführte Begriff der Redundanz erlaubt auch noch eine andere Vorgehensweise. Bei der Vervollständigung einer Klauselmenge S bezüglich eines beliebigen Inferenzsystems dürfen jederzeit zusätzlich Klauseln, die aus S folgen, zu S hingenommen werden, ohne die Korrektheit oder Vollständigkeit des Verfahrens zu verlieren. Daher ist es teilweise möglich, geschickt eine Klausel C', die aus S folgt, so hinzuzunehmen, dass eine Klausel C aus S dadurch redundant wird und damit entfernt werden kann. Aufgrund der Definition von Redundanz ist dabei C' im Allgemeinen einfacher als C. Man sagt daher C kann zu C' *simplifiziert* werden. Ein weiteres Ziel muss es also sein, geeignete *Simplifikationsregeln* zu entwickeln, das heißt Regeln, wann eine Klausel simplifiziert werden kann.

Sowohl Redundanzkriterien als auch Simplifikationsregeln stellen eine praktische Umsetzung der Theorie dar, die für die Effizienz einer Implementierung entscheidend ist. Welche Kriterien und Regeln sinnvoll sind, muss durch Experimente gezeigt werden.

5.1. Implementierung

Für die Implementierung der entwickelten Inferenzsysteme wurde das System PRAKSYS verwendet. Dabei handelt es sich um einen im Rahmen eines Praktikums der AG Avenhaus an der Universität Kaiserslautern im Wintersemester 2001/2002 entwickelten Theorembeweiser für die reine Gleichheitslogik erster Stufe. Entwicklungsziel von *Praksys* war es nicht, einen möglichst schnellen Beweiser zu entwickeln, sondern eine Umgebung zur Verfügung zu stellen, die das Testen und den Vergleich von verschiedenen Beweisverfahren erlaubt. Daher ist Praksys sehr leicht und umfangreich konfigurierbar und besitzt eine ausführliche Statistikfunktion.

Im Wesentlichen besteht Praksys aus einem Framework, das es erlaubt, unterschiedliche Implementierungen für die einzelnen Komponenten zu verwenden. Die zentrale Komponente des Systems ist die sogenannte *Hauptschleife*. Diese Komponente besitzt die Aufgabe, zu einer übergebenen *Spezifikation* einen Beweis zu finden. Diese Spezifikation enthält als Hauptbestandteil eine Menge von Axiomen und eine aus diesen Axiomen zu folgernde Konklusion. Die Komponenten, die die Hauptschleife zur Beweissuche verwendet, sind genau wie die Hauptschleife selbst in ihrer Implementierung austauschbar. Das Framework bietet außerdem Funktionen zur Erfassung von statistischen Daten über die Programmläufe an.

Aufgrund dieser Eigenschaften eignete sich Praksys gut als Grundlage für die Umsetzung und den Vergleich der Varianten des Inferenzsystems. Es stellte grundlegende Objekte, wie Terme, Gleichungen, Klauseln, Regeln, Reduktionssysteme usw. bereit. Außerdem waren grundlegende Algorithmen wie z. B. ein Unifikationsalgorithmus bereits vorhanden. Zur Umsetzung der verschiedenen Varianten des Inferenzsystems mussten Datenstrukturen zur Darstellung von Constraints und Funktionen auf diesen, z. B. für die Vereinfachung von Constraints, den Test auf Erfüllbarkeit oder den Vergleich zweier Klauseln mit Constraint auf syntaktische Äquivalenz bis auf Variablenumbenennung, implementiert werden. Zusätzlich war die Implementierung einer neuen Hauptschleife erforderlich, da die vorhandene Hauptschleife nicht in ausreichendem Maße konfigurierbar war.

Eine Einschränkung bei der Verwendung von Praksys ist, dass nur sogenannte *Unitklauseln* unterstützt werden. Unitklauseln sind spezielle Hornklauseln, nämlich Klauseln, die aus nur einem Literal bestehen. Der Hauptgrund für die Entscheidung, Praksys nicht auf allgemeine Hornklauseln zu erweitern, besteht darin, dass hierzu umfangreiche Anpassungen an der grundlegenden Datenstrukturen und Algorithmen nötig sind. Hinzu kommt, dass sich bei der Betrachtung von Unitklauseln leichter Redundanzkriterien und Simplifikationsregeln finden lassen als bei der Betrachtung allgemeiner Hornklauseln. Außerdem entfällt für Unitklauseln die Fragestellung, welche Literale selektiert werden sollten, da das einzige evtl. vorhandene negative Literal nach Selektionsregel immer selektiert ist. Da Unitklauseln nur eine spezielle, einfache Form von Hornklauseln sind, müssen die bisherigen Überlegungen nicht angepasst werden. Da sich die Inferenzregeln jedoch teilweise stark vereinfachen, soll das Inferenzsystem $\mathcal{H}_\mathcal{C}$ noch einmal speziell für Unitklauseln und in einer Variante, bei der die entstehenden Constraints nur auf Erfüllbarkeit überprüft und anschließend aufgegeben werden, aufgestellt werden. Außerdem sollen für diese neuen Inferenzsysteme Redundanzkriterien und Simplifikationsregeln entwickelt werden.

5.2. Das Inferenzsystem $\mathcal{H}_{\mathcal{CU}}$

Das folgende Inferenzsystem $\mathcal{H}_{\mathcal{CU}}$ entspricht dem Inferenzsystem $\mathcal{H}_{\mathcal{C}}$ speziell für Unitklauseln mit Constraint.

Superposition:

$$\frac{l \simeq r \mid \aleph \qquad s \mathrel{\dot{\simeq}} t \mid \beth}{s[r]_p \mathrel{\dot{\simeq}} t \mid s|_p \doteq l \;\wedge\; l > r \;\wedge\; s > t \;\wedge\; s \mathrel{\dot{\simeq}} t > l \simeq r \;\wedge\; \aleph \;\wedge\; \beth}$$

falls

- ∘ der Constraint der Konklusion ist erfüllbar
- ∘ $s|_p$ ist keine Variable

Reflexivitäts-Resolution:

$$\frac{s \not\simeq t \mid \aleph}{\square}$$

falls

- der Constraint $s \doteq t \;\wedge\; \aleph$ ist erfüllbar

Bemerkung 5 Man beachte, dass bei der Reflexivitäts-Resolution nun die Bedingung, dass der Constraint der Konklusion erfüllbar ist, für die Korrektheit nötig ist. Dies liegt daran, dass nach Definition $\square$ immer unerfüllbar sein und somit einen erfüllbaren Constraint besitzen muss. Das Problem die leere Klausel zu erkennen trat schon früher auf, wurde dort aber nicht im Inferenzsystem offensichtlich.

Es wird also ein Verfahren benötigt, dass entscheiden kann, ob der Constraint erfüllbar ist. Wie in Abschnitt 3.3 vorgestellt, existieren solche Verfahren zwar, sollen hier aber nicht näher betrachtet werden. Eine Lösung des Problems ist es, nur den Gleichheitsanteil des Constraints zu betrachten. Die Ordnungsbedingungen können jederzeit aufgegeben werden, da sie nur der Beschränkung des Suchraums dienen. Außerdem kann die Erfüllbarkeit eines reinen Gleichheitsconstraints leicht bestimmt werden, da ein reiner Gleichheitsconstraint nämlich genau dann unerfüllbar ist, wenn er keinen Unifikator besitzt.

Bemerkung 6 Auch hier kann der bei der *Superposition* auftretende Constraint $s|_p \doteq l \;\wedge\; l > r \;\wedge\; s > t \;\wedge\; s \mathrel{\dot{\simeq}} t > l \simeq r$ leicht vereinfacht werden.

Ist $s \mathrel{\dot{\simeq}} t$ eine Ungleichung oder gilt $p \neq \lambda$, so wird $s \mathrel{\dot{\simeq}} t > l \simeq r$ von jeder Substitution erfüllt, die den Constraint $s|_p \doteq l \;\wedge\; l > r \;\wedge\; s > t$ erfüllt, und kann somit weggelassen werden. Ist hingegen $s \mathrel{\dot{\simeq}} t$ eine Gleichung und gilt $p = \lambda$, so erfüllt eine Substitution, die $s|_p \doteq l$ erfüllt, genau dann $s \mathrel{\dot{\simeq}} t > l \simeq r$, wenn sie den Constraint $t > r$ erfüllt. Daher kann in diesem Fall $s \mathrel{\dot{\simeq}} t > l \simeq r$ zu $t > r$ vereinfacht werden.

Satz 5 Sei S_0 eine Menge von Unitklauseln ohne Constraint, das heißt eine Menge von Unitklauseln mit allgemeingültigem Constraint $\top$. Weiter sei S_A ein Abschluss bis auf Redundanz der Menge S_0 bezüglich $\mathcal{H}_{\mathcal{CU}}$. Dann ist S_0 genau dann erfüllbar, wenn S_A nicht die leere Klausel enthält.

Beweis Dies ist ein Spezialfall von Satz 4.

5.3. Redundanzkriterien und Simplifikationsregeln für $\mathcal{H}_{\mathcal{CU}}$

Im Folgenden sollen Redundanzkriterien und Simplifikationsregeln speziell für das Inferenzsystem $\mathcal{H}_{\mathcal{CU}}$, also für Unitklauseln mit Constraint entwickelt werden. Dazu wird vorher aber noch folgende Definition benötigt:

Definition 2 Eine Klausel $D \mid \beth$ ist *relativ reduziert* modulo einer Substitution η bezüglich einer Klausel $C \mid \aleph$ genau dann, wenn für alle grundkonvergenten Reduktionssysteme R und für alle Substitutionen σ gilt: Sind $C{\cdot}\sigma$ und $D{\cdot}\eta\sigma$ Grundinstanzen von $C \mid \aleph$ und $D \mid \beth$ und ist $C{\cdot}\sigma$ an Substitutionspositionen nicht reduzierbar bezüglich R, so ist auch $D{\cdot}\eta\sigma$ nicht reduzierbar an Substitutionspositionen bezüglich R.

Beispiel Im Folgenden soll zur Verdeutlichung dieser Definition gezeigt werden, dass die Klausel $D \mid \beth \equiv (x \simeq g(y, f(y))) \mid x \doteq a$ relativ reduziert modulo $\eta = \{y \leftarrow f(z')\}$ bezüglich $C \mid \aleph \equiv (x' \simeq g(y', a)) \mid x' \simeq f(a) \ \wedge \ y' \simeq f(z')$ ist.

Sei σ eine beliebige Substitutition, für die $C{\cdot}\sigma$ und $D{\cdot}\eta\sigma$ Grundinstanzen von $C \mid \aleph$ und $D \mid \beth$ sind, und R ein beliebiges grundkonvergentes Reduktionssystem. Dann kommen die Terme $a\sigma \equiv a$ und $f(z')\sigma$ an Substitutionspositionen von $D{\cdot}\eta\sigma$ vor und die Terme $x'\sigma$ und $y'\sigma$ an Substitutionspositionen von $C{\cdot}\sigma$. Zu zeigen ist also, dass wenn $x'\sigma$ und $y'\sigma$ nicht reduzierbar bezüglich R sind, auch a und $f(z')\sigma$ nicht reduzierbar bezüglich R sind.

Sei $\sigma_\aleph = mgu(eqpart(\aleph))$. Dann gilt $x'\sigma_\aleph \equiv f(a)\sigma_\aleph$ und $y'\sigma_\aleph \equiv f(z')\sigma_\aleph$. Außerdem existiert eine Substitution $\theta_\aleph$ mit $\sigma = \theta_\aleph \circ \sigma_\aleph$. Damit gilt $x'\sigma \equiv x'\sigma_\aleph\theta_\aleph \equiv f(a)\sigma_\aleph\theta_\aleph \equiv f(a)$. Somit ist a nicht reduzierbar bezüglich R, da $x'\sigma_\aleph\theta_\aleph \equiv f(a)$ nicht reduzierbar bezüglich R ist. Außerdem gilt der Zusammenhang $y'\sigma \equiv y'\sigma_\aleph\theta_\aleph \equiv f(z')\sigma_\aleph\theta_\aleph \equiv f(z')\sigma$. Also ist $f(z')\sigma$ nicht reduzierbar bezüglich R, da $y'\sigma$ nicht reduzierbar bezüglich R ist.

Somit ist $D \mid \beth$ wirklich relativ reduziert modulo η bezüglich $C \mid \aleph$.

Es ist leider sehr aufwändig nachzuprüfen, ob eine Klausel relativ reduziert modulo einer Substitution bezüglich einer anderen Klausel ist. Deswegen ist es sinnvoll, bereits für diesen Test ein hinreichendes Kriterium zu verwenden. Dazu soll die im Beispiel verwendete Vorgehensweise formalisiert werden:

Lemma 26 Seien $C \mid \aleph$ und $D \mid \beth$ zwei Klauseln und η eine Substitution. Sei $\sigma_\aleph = mgu(eqpart(\aleph))$ und $\sigma_\beth = mgu(eqpart(\beth))$ mit $\sigma_\aleph$ und $\sigma_\beth$ idempotent. Existiert für alle $x \in var(D)$ ein $y \in var(C)$ mit $x\sigma_\beth\eta$ ist Teilterm von $y\sigma_\aleph$, so ist $D \mid \beth$ relativ reduziert modulo η bezüglich $C \mid \aleph$.

Beweis Sei R ein beliebiges grundkonvergentes Reduktionssystem und σ eine beliebige Substitution, für die $C{\cdot}\sigma$ und $D{\cdot}\eta\sigma$ Grundinstanzen von $C \mid \aleph$ und $D \mid \beth$ sind und für die $C{\cdot}\sigma$ an Substitutionspositionen nicht reduzierbar bezüglich R ist. Zu zeigen ist dann, dass $D{\cdot}\eta\sigma$ nicht reduzierbar an Substitutionspositionen bezüglich R ist.

Angenommen $D{\cdot}\eta\sigma$ ist reduzierbar an Substitutionspositionen bezüglich R. Dann gibt es eine Variable $x \in var(D)$, so dass $x\eta\sigma$ keine Variable und reduzierbar bezüglich R ist. Da $D{\cdot}\eta\sigma$ Grundinstanz von $D \mid \beth$ ist, erfüllt die Substitution $\sigma \circ \eta$ den Constraint $\beth$ und ist somit Instanz von $\sigma_\beth$, das heißt es existiert eine Substitution $\theta_\beth$ mit $\sigma \circ \eta = \theta_\beth \circ \sigma_\beth$. Da $C{\cdot}\sigma$ Grundinstanz von $C \mid \aleph$ ist, existiert ebenso eine Substitution $\theta_\aleph$ mit $\sigma = \theta_\aleph \circ \sigma_\aleph$.

Nach Voraussetzung gibt es eine Variable $y \in var(C)$ mit $x\sigma_\beth\eta$ ist Teilterm von $y\sigma_\aleph$. Damit ist $x\sigma_\beth\eta\sigma$ ein Teilterm von $y\sigma_\aleph\sigma$. Es gilt $x\sigma_\beth\eta\sigma \equiv x\sigma_\beth\sigma_\beth\theta_\beth \equiv x\sigma_\beth\theta_\beth \equiv x\eta\sigma$ und $y\sigma_\aleph\sigma \equiv y\sigma_\aleph\sigma_\aleph\theta_\aleph \equiv y\sigma_\aleph\theta_\aleph \equiv y\sigma$. Da $x\eta\sigma$ also Teilterm von $y\sigma$ und $x\eta\sigma$ keine Variable und reduzierbar bezüglich R ist, ist $y\sigma$ keine Variable und reduzierbar bezüglich R. Dies ist wegen $y \in var(C)$ aber ein Widerspruch dazu, dass $C{\cdot}\sigma$ an Substitutionspositionen nicht reduzierbar bezüglich R ist. □

Beispiel Wie oben gezeigt ist $D \mid \beth \equiv (x \simeq g(y, f(y))) \mid x \doteq a$ relativ reduziert modulo $\eta = \{y \leftarrow f(z')\}$ bezüglich $C \mid \aleph \equiv (x' \simeq g(y', a))) \mid x' \simeq f(a) \wedge y' \simeq f(z')$. Dieses Beispiel soll hier erneut aufgegriffen werden und diesmal mit Hilfe von Lemma 26 behandelt werden.

Es gilt $\sigma_\aleph = \{x' \leftarrow f(a), y' \leftarrow f(z')\}$ und $\sigma_\beth = \{x \leftarrow a\}$. Somit ist $x\sigma_\beth\eta \equiv a$ Teilterm von $x'\sigma_\aleph \equiv f(a)$ und $y\sigma_\beth\eta \equiv f(z')$ Teilterm von $y'\sigma_\aleph \equiv f(z')$. Also kann Lemma 26 angewendet werden.

5.3.1. Klauseln mit unerfüllbarem Constraint

Satz 6 Eine Klausel $C \mid \aleph \in S$, wobei $\aleph$ ein unerfüllbarer Constraint ist, ist redundant bezüglich S.

Beweis Da $\aleph$ unerfüllbar ist, besitzt $C \mid \aleph$ keine Grundinstanzen. Damit ist jede Grundinstanz redundant bezüglich S und damit ist auch $C \mid \aleph$ redundant bezüglich S. □

5.3.2. Tautologien

Satz 7 Sei $C \mid \aleph \in S$ eine Klausel der Form $s \simeq t \mid \aleph$ und $\sigma_\aleph = mgu(eqpart(\aleph))$. Gilt $s\sigma_\aleph \equiv t\sigma_\aleph$, so ist $S \mid \aleph$ redundant bezüglich S.

Beweis Sei $C{\cdot}\sigma$ eine beliebige Grundinstanz von $C \mid \aleph$. Damit erfüllt σ den Constraint $\aleph$ und ist somit Instanz von $\sigma_\aleph$. Damit gilt dann aber $s\sigma \equiv t\sigma$. Also ist $C{\cdot}\sigma \equiv (s \simeq t){\cdot}\sigma$ bezüglich der leeren Menge redundant. Damit ist $C \mid \aleph$ redundant bezüglich S. □

5.3.3. Subsumption

Satz 8 Seien $C \mid \aleph$ und $D \mid \beth$ zwei Klauseln aus S. Existiert eine Substitution η mit

- $D\eta \equiv C$
- $SubPos(D{\cdot}\eta) \neq \emptyset$
- für jede Substitution σ, die $\aleph$ erfüllt, erfüllt $\sigma \circ \eta$ den Constraint $\beth$
- $D \mid \beth$ ist relativ reduziert modulo η zu $C \mid \aleph$

so ist $C \mid \aleph$ redundant bezüglich S. Man sagt dann $D \mid \beth$ subsumiert $C \mid \aleph$.

Beweis Sei $C{\cdot}\sigma$ eine beliebige Grundinstanz von $C \mid \aleph$. Da $C{\cdot}\sigma$ Grundinstanz von $C \mid \aleph$ ist, erfüllt σ den Constraint $\aleph$, und nach Voraussetzung erfüllt somit $\sigma \circ \eta$ den Constraint $\beth$. Damit ist $D{\cdot}\eta\sigma$ Grundinstanz von $D \mid \beth$. Im Folgenden soll gezeigt werden, dass $C{\cdot}\sigma$ redundant bezüglich $D{\cdot}\eta\sigma$ ist.

Wegen $C \equiv D\eta$ gilt $C\sigma \equiv D\eta\sigma$. Damit gilt für alle grundkonvergenten Reduktionssysteme R trivialerweise $R^* \models C\sigma$, falls $R^* \models D\eta\sigma$. Da $SubPos(D{\cdot}\eta) \neq \emptyset$ und $C \equiv D\eta$ gilt, gilt $SubPos(D{\cdot}\eta\sigma) \supset SubPos(C{\cdot}\sigma)$. Zusammen mit $D\eta\sigma \equiv C\sigma$ folgt $C{\cdot}\sigma \succ D{\cdot}\eta\sigma$. Da $D \mid \beth$ relativ reduziert modulo η zu $C \mid \aleph$ ist und $C{\cdot}\sigma$ und $D{\cdot}\eta\sigma$ Grundinstanzen von $C \mid \aleph$ und $D \mid \beth$ sind, ist $D{\cdot}\eta\sigma$ bezüglich jedes beliebigen grundkonvergenten Reduktionssystems R nicht reduzierbar an Substitutionspositionen, falls $C{\cdot}\sigma$ nicht reduzierbar an Substitutionspositionen bezüglich R ist.

Also ist $C{\cdot}\sigma$ wirklich redundant bezüglich $D{\cdot}\eta\sigma$ und ist damit redundant bezüglich S. Da $C{\cdot}\sigma$ als beliebige Grundinstanz von $C \mid \aleph$ gewählt wurde, ist somit $C \mid \aleph$ redundant bezüglich S. □

Bemerkung 7 Dieses Konzept lässt sich noch verstärken. Seien $C \mid \aleph$ und $D \mid \beth$ zwei Klauseln aus S und C besitze die Form $u[s]_p \simeq u[t]_p$. Dann ist $C \mid \aleph$ redundant bezüglich S, falls $D \mid \beth$ die Klausel $s \simeq t \mid \aleph$ subsumiert.

5.3.4. Variablen-Abstraktion

Satz 9 Sei $C \mid \aleph \in S$ eine Klausel und p eine Position von C mit $C|_p$ ist keine Variable. Sei y eine neue Variable, also eine Variable, die in $C \mid \aleph$ nicht vorkommt. Ist $C[y]_p \mid y \doteq C|_p \ \wedge \ \aleph$ relativ reduziert modulo id bezüglich $C \mid \aleph$, so kann $C \mid \aleph$ zu $C[y]_p \mid y \doteq C|_p \ \wedge \ \aleph$ simplifiziert werden.

Beweis Um zu zeigen, dass $C \mid \aleph$ zu $C[y]_p \mid y \doteq C|_p \ \wedge \ \aleph$ simplifiziert werden kann, muss gezeigt werden, dass $C[y]_p \mid y \doteq C|_p \ \wedge \ \aleph$ eine logische Folgerung aus S ist und dass $C \mid \aleph$ in $S \cup \{C[y]_p \mid y \doteq C|_p \ \wedge \ \aleph\}$ redundant ist.

Jede Grundinstanz von $C[y]_p \mid y \doteq C|_p \wedge \aleph$ ist auch Grundinstanz von $C \mid \aleph \in S$. Somit folgt $C[y]_p \mid y \doteq C|_p \wedge \aleph$ logisch aus S. Mit $\eta := \{y \leftarrow C|_p\}$ gilt offensichtlich $C[y]_p\eta \equiv C$ und $SubPos(C[y]_p{\cdot}\eta) = \{p\} \neq \emptyset$. Außerdem ist nach Voraussetzung $C[y]_p \mid y \doteq C|_p \ \wedge \ \aleph$ relativ reduziert modulo id bezüglich $C \mid \aleph$ und daher auch relativ reduziert modulo η, da für jede Substitution σ, die $y \doteq C|_p \ \wedge \ \aleph$ erfüllt, $\eta\sigma \equiv id\sigma$ gilt. Also subsumiert $C[y]_p \mid y \doteq C|_p \ \wedge \ \aleph$ die Klausel $C \mid \aleph$ unter Verwendung von η. □

Beispiel Die Klausel $C \mid \aleph := f(f(a,x),a) \simeq y \mid y \doteq f(a,x)$ kann mittels Variablenabstraktion zu $D \mid \beth := f(z,a) \simeq y \mid y \doteq f(a,x) \ \wedge \ z \doteq f(a,x)$ simplifiziert werden. Denn $D \mid \beth$ ist relativ reduziert modulo id bezüglich $C \mid \aleph$, da für alle Substitutionen σ, die $\aleph$ und $\beth$ erfüllen, die Mengen aller Terme, die an Substitutionspositionen von $C{\cdot}\sigma$ und $D{\cdot}id\sigma$ auftreten, übereinstimmen. Die Klausel $D \mid \beth$ kann dann weiter zu $f(z,v) \simeq y \mid y \doteq f(a,x) \ \wedge \ z \doteq f(a,x) \ \wedge \ v \doteq a$ simplifiziert werden. Diese Klausel ist bis auf Variablenumbenennung äquivalent zu $f(y,v) \simeq y \mid y \doteq f(a,x) \ \wedge \ v \doteq a$.

5.3.5. Reduktions-Simplifikation

Satz 10 Seien $C \mid \aleph$ und $D \mid \beth \equiv l \simeq r \mid \beth$ zwei Klauseln aus S. Existiert eine Position p und eine Substitution η mit

- $C|_p \equiv l\eta$
- $l\eta \succ r\eta$
- $C \succ D\eta$
- für jede Substitution θ, die $\aleph$ erfüllt, erfüllt $\theta \circ \eta$ den Constraint $\beth$
- $D \mid \beth$ ist relativ reduziert modulo η bezüglich $C \mid \aleph$

so kann $C \mid \aleph \equiv C[l\eta]_p \mid \aleph$ zu $C[r\eta]_p \mid \aleph$ simplifiziert werden.

Beweis Um zu zeigen, dass $C \mid \aleph$ zu $C[r\eta]_p \mid \aleph$ simplifiziert werden kann, muss gezeigt werden, dass $C[r\eta]_p \mid \aleph$ eine logische Folgerung aus S ist und dass $C \mid \aleph$ bezüglich $S \cup \{C[r\eta]_p \mid \aleph\}$ redundant ist.

Zunächst soll gezeigt werden, dass $C[r\eta]_p \mid \aleph$ eine logische Folgerung aus S ist. Sei $C[r\eta]_p{\cdot}\sigma$ eine beliebige Grundinstanz von $C[r\eta]_p \mid \aleph$. Der Constraint $\aleph$ wird daher von σ erfüllt. Nach Voraussetzung erfüllt dann $\sigma \circ \eta$ den Constraint $\beth$ und somit ist $D{\cdot}\eta\sigma$ eine Instanz von $D \mid \beth$. Außerdem ist $C{\cdot}\sigma$ eine Instanz von $C \mid \aleph$. O. B. d. A. seien $D{\cdot}\eta\sigma$ und $C{\cdot}\sigma$ Grundinstanzen. Offensichtlich folgt $C[r\eta]_p{\cdot}\sigma$ aus $C{\cdot}\sigma$ und $D{\cdot}\eta\sigma \equiv l \simeq r{\cdot}\eta\sigma$. Damit folgt $C[r\eta]_p \mid \aleph$ aus S.

Zu zeigen bleibt, dass $C \mid \aleph$ bezüglich $S \cup \{C[r\eta]_p \mid \aleph\}$ redundant ist. Sei $C{\cdot}\sigma$ eine beliebige Grundinstanz von $C \mid \aleph$. Der Constraint $\aleph$ wird daher von σ erfüllt. Nach Voraussetzung erfüllt dann $\sigma \circ \eta$ den Constraint $\beth$. Wegen $C|_p \equiv l\eta$ gilt $var(C) \supseteq var(l\eta)$ und wegen $l\eta \succ r\eta$ gilt $var(l\eta) \supseteq var(r\eta)$. Damit sind $C[r\eta]_p{\cdot}\sigma$ und $D{\cdot}\eta\sigma$ Grundinstanzen von $C[r\eta]_p \mid \aleph$ und $D \mid \beth$. Dann ist $C{\cdot}\sigma$ redundant bezüglich $D{\cdot}\eta\sigma \equiv (l \simeq r){\cdot}\eta\sigma$ und $C[r\eta]_p{\cdot}\sigma$, denn:

- Gilt in einem beliebigen grundkonvergenten Reduktionssystem $R^* \models (l \simeq r){\cdot}\eta\sigma$ und $R^* \models C[r\eta]_p{\cdot}\sigma$, so gilt auch $R^* \models C[l\eta]_p{\cdot}\sigma$.
- Es gilt $C{\cdot}\sigma \succ C[r\eta]_p{\cdot}\sigma$, da wegen $l\eta \succ r\eta$ die Beziehung $C \equiv C[l\eta]_p \succ C[r\eta]_p$ gilt. Da $C \succ D\eta$ nach Voraussetzung gilt, ist $C{\cdot}\sigma \succ D{\cdot}\eta\sigma$.
- Nach Voraussetzung ist $D \mid \beth$ relativ reduziert bezüglich $C \mid \aleph$ modulo η. Damit ist $D{\cdot}\eta\sigma$ nicht reduzierbar an Substitutionspositionen bezüglich jedes beliebigen grundkonvergenten Reduktionssystems R, falls $C{\cdot}\sigma$ nicht reduzierbar an Substitutionspositionen bezüglich R ist. Wegen $var(C[r\eta]_p) \subseteq var(C)$ ist in diesem Fall ebenfalls $C[r\eta]_p{\cdot}\sigma$ nicht reduzierbar an Substitutionspositionen bezüglich R.

Also ist $C\cdot\sigma$ redundant bezüglich $S \cup \{C[r\eta]_p \mid \aleph\}$. Somit ist auch $C \mid \aleph$ redundant bezüglich $S \cup \{C[r\eta]_p \mid \aleph\}$. $\square$

5.4. Das Inferenzsystem $\mathcal{H}_{\mathcal{U}}$

Das folgende Inferenzsystem $\mathcal{H}_{\mathcal{U}}$ entspricht dem Inferenzsystem $\mathcal{H}_{\mathcal{CU}}$, wobei alle Constraints direkt nach dem Auftreten auf Erfüllbarkeit geprüft und dann aufgegeben werden.

Superposition:

$$\frac{l \simeq r \qquad s \mathrel{\dot{\simeq}} t}{(s[r]_p \mathrel{\dot{\simeq}} t)\sigma}$$

falls

- $\sigma = mgu(s|_p, l)$
- ◦ der Constraint $(l > r \ \wedge \ s > t \ \wedge \ s \mathrel{\dot{\simeq}} t > l \simeq r)\sigma$ ist erfüllbar
- ◦ $s|_p$ ist keine Variable

Reflexivitäts-Resolution:

$$\frac{s \not\simeq t}{\square}$$

falls

- s und t sind unifizierbar

Satz 11 Sei S_0 eine Menge von Unitklauseln ohne Constraint und S_A ein Abschluss bis auf Redundanz der Menge S_0 bezüglich $\mathcal{H}_{\mathcal{U}}$. Dann ist S_0 genau dann erfüllbar, wenn S_A nicht die leere Klausel enthält.

Beweis Dies ist ein Spezialfall von Satz 4.

5.5. Redundanzkriterien und Simplifikationsregeln für $\mathcal{H}_{\mathcal{U}}$

Im Folgenden sollen Redundanzkriterien und Simplifikationsregeln speziell für das Inferenzsystem $\mathcal{H}_{\mathcal{U}}$, also für Unitklauseln ohne Constraint entwickelt werden.

Lemma 27 Eine Grundunitklausel $C\cdot\sigma$ ist redundant bezüglich einer Menge S von Unitklauseln ohne Constraint, falls es Grundinstanzen $D_1\cdot\theta, \ldots, D_n\cdot\sigma$ von Klauseln $D_1, \ldots, D_n$ aus S gibt, so dass für alle grundkonvergenten Reduktionssysteme R gilt:

- Gilt $R^* \models D_1\cdot\theta, \ldots D_n\cdot\theta$, so gilt auch $R^* \models C\cdot\sigma$.
- $C\cdot\sigma \succ D_i\cdot\theta$ für $1 \leq i \leq n$

Beweis Sei R ein beliebiges grundkonvergentes Reduktionssystem und $x\theta' := x\theta\Downarrow_R$ für alle $x \in dom(\theta)$. Für alle $1 \leq i \leq n$ ist $D_i\cdot\theta'$ eine Grundinstanz von D_i, da D_i keinen Constraint besitzt. Dann gilt:

- Gilt $R^* \models D_1{\cdot}\theta', \ldots D_n{\cdot}\theta'$, so gilt auch $R^* \models D_1{\cdot}\theta, \ldots D_n{\cdot}\theta$ nach Definition von θ'. Nach Voraussetzung gilt damit auch $R^* \models C{\cdot}\sigma$.
- $C{\cdot}\sigma \succ D_i{\cdot}\theta \succ D_i{\cdot}\theta'$ für $1 \leq i \leq n$
- $D_1{\cdot}\theta', \ldots, D_n{\cdot}\theta'$ sind nicht reduzierbar an Substitutionspositionen bezüglich R.

Damit ist $C{\cdot}\sigma$ redundant bezüglich $D_1{\cdot}\theta', \ldots D_n{\cdot}\theta'$ und bezüglich R. Da R beliebig gewählt war, ist damit $C{\cdot}\sigma$ redundant bezüglich S. □

5.5.1. Tautologien

Satz 12 Sei $C \in S$ eine Klausel der Form $s \simeq s$. Dann ist C redundant bezüglich S.

Beweis Sei $C{\cdot}\sigma$ eine beliebige Grundinstanz von C. $C{\cdot}\sigma \equiv (s \simeq s){\cdot}\sigma$ ist bezüglich der leeren Menge redundant. Damit ist C redundant bezüglich S. □

5.5.2. Subsumption

Satz 13 Seien C und D zwei Klauseln aus S. Existiert eine Substitution η mit $D\eta \equiv C$ und $SubPos(D{\cdot}\eta) \neq \emptyset$ so ist C redundant bezüglich S.

Beweis Sei $C{\cdot}\sigma$ eine beliebige Grundinstanz von C. Damit ist $D{\cdot}\eta\sigma$ Grundinstanz von D. Im Folgenden soll gezeigt werden, dass $C{\cdot}\sigma$ redundant bezüglich $D{\cdot}\eta\sigma$ ist.

Wegen $C \equiv D\eta$ gilt $C\sigma \equiv D\eta\sigma$. Damit gilt für alle grundkonvergenten Reduktionssysteme R trivialerweise $R^* \models C\sigma$, falls $R^* \models D\eta\sigma$. Da $SubPos(D{\cdot}\eta) \neq \emptyset$ und $C \equiv D\eta$ gilt, gilt $SubPos(D{\cdot}\eta\sigma) \supset SubPos(C{\cdot}\sigma)$. Zusammen mit $D\eta\sigma \equiv C\sigma$ folgt $C{\cdot}\sigma \succ D{\cdot}\eta\sigma$.

Nach Lemma 27 ist $C{\cdot}\sigma$ redundant bezüglich S. Da $C{\cdot}\sigma$ als beliebige Grundinstanz von C gewählt wurde, ist somit C redundant bezüglich S. □

5.5.3. Reduktions-Simplifikation

Satz 14 Seien C und $D \equiv l \simeq r$ zwei Klauseln aus S. Existiert eine Position p und eine Substitution η mit

- $C|_p \equiv l\eta$
- $l\eta \succ r\eta$
- $C \succ D\eta$

so kann $C \equiv C[l\eta]_p$ zu $C[r\eta]_p$ simplifiziert werden.

Beweis Um zu zeigen, dass C zu $C[r\eta]_p$ simplifiziert werden kann, muss gezeigt werden, dass $C[r\eta]_p$ eine logische Folgerung aus S ist und dass C bezüglich $S \cup \{C[r\eta]_p\}$ redundant ist.

Zunächst soll gezeigt werden, dass $C[r\eta]_p$ eine logische Folgerung aus S ist. Sei $C[r\eta]_p{\cdot}\sigma$ eine beliebige Grundinstanz von $C[r\eta]_p$. O. B. d. A. seien $D{\cdot}\eta\sigma$ und $C{\cdot}\sigma$

Grundinstanzen von D und C. Offensichtlich folgt $C[r\eta]_p{\cdot}\sigma$ aus $C{\cdot}\sigma$ und $D{\cdot}\eta\sigma \equiv (l \simeq r){\cdot}\eta\sigma$. Damit folgt $C[r\eta]_p$ aus S.

Zu zeigen bleibt, dass C bezüglich $S \cup \{C[r\eta]_p\}$ redundant ist. Sei $C{\cdot}\sigma$ eine beliebige Grundinstanz von C. Wegen $C|_p \equiv l\eta$ gilt $var(C) \supseteq var(l\eta)$ und wegen $l\eta \succ r\eta$ gilt $var(l\eta) \supseteq var(r\eta)$. Damit sind $C[r\eta]_p{\cdot}\sigma$ und $D{\cdot}\eta\sigma$ Grundinstanzen von $C[r\eta]_p$ und D. Nach Lemma 27 ist dann $C{\cdot}\sigma$ redundant bezüglich $S \cup \{C[r\eta]_p\}$, denn:

- Gilt in einem beliebigen grundkonvergenten Reduktionssystem $R^* \models (l \simeq r){\cdot}\eta\sigma$ und $R^* \models C[r\eta]_p{\cdot}\sigma$, so gilt auch $R^* \models C[l\eta]_p{\cdot}\sigma$.

- Es gilt $C{\cdot}\sigma \succ C[r\eta]_p{\cdot}\sigma$, da wegen $l\eta \succ r\eta$ die Beziehung $C \equiv C[l\eta]_p \succ C[r\eta]_p$ gilt. Da $C \succ D\eta$ nach Voraussetzung gilt, ist $C{\cdot}\sigma \succ D{\cdot}\eta\sigma$.

Da $C{\cdot}\sigma$ als beliebige Grundinstanz von C gewählt wurde, ist somit auch C redundant bezüglich $S \cup \{C[r\eta]_p \mid \aleph\}$. □

5.6. Hinreichender Unerfüllbarkeitstest für Constraints

Die Frage, ob ein allgemeiner Constraint erfüllbar ist, ist nicht leicht zu beantworten. Besteht der Constraint nur aus atomaren Gleichheitsconstraints, so handelt es sich dabei um ein reines Unifikationsproblem, das mit den bekannten Methoden gelöst werden kann. Verwendet man jedoch zusätzlich Ordnungsconstraints, ist die Antwort zum einen abhängig von der Reduktionsordnung, die für die Interpretation der Ordnungsconstraints verwendet wird und zum anderen als $\mathcal{NP}$-hart nachgewiesen (für die LPO z. B. in: [Nie93]). Es ist jedoch an vielen Stellen im System sinnvoll, Constraints auf Erfüllbarkeit zu prüfen. Zum einen gibt es einige Inferenzregeln mit Erfüllbarkeitsbedingungen für Constraints und zum anderen ein Redundanzkriterium, das besagt, dass Klauseln mit unerfüllbarem Constraint keine Grundinstanzen repräsentieren und somit nicht weiter betrachtet werden müssen (siehe Abschnitt 5.3.1). Um dies zu ermöglichen, ist es nützlich, möglichst effizient und unabhängig von der verwendeten Reduktionsordnung unerfüllbare Constraints zu erkennen. Ein Entscheidungsverfahren ist jedoch nicht notwendig, da es durchaus erlaubt ist, Klauseln mit unerfüllbarem Constraint beizubehalten. Für die Reflexivitäts-Resolution, für die ein Entscheidungsverfahren notwendig wäre, reicht es (wie in Bemerkung 5 erläutert) den Gleichheitsanteil des Constraints zu prüfen.

Es gibt einige hinreichende Kriterien, mit denen effizient die Unerfüllbarkeit eines Constraints nachgewiesen werden kann. In wenigen Fällen kann damit auch die Erfüllbarkeit nachgewiesen werden, man erhält jedoch kein Entscheidungsverfahren. Diese Kriterien beruhen auf den Eigenschaften, die alle Reduktionsordnungen nach Definition haben müssen. Dazu zählen die in Abschnitt 1.5 vorgestellten allgemeinen Eigenschaften von Partialordnungen (Irreflexivität, Antisymmetrie und Transitivität) sowie die speziellen Eigenschaften von Reduktionsordnungen (insbesondere Verträglichkeit mit der Termstruktur).

Diese Kriterien können dazu genutzt werden, die Unerfüllbarkeit von Constraints nachzuweisen (siehe [Sch04]), aber auch dazu, Constraints zu vereinfachen, indem atomare Constraints, die immer erfüllt sind oder die aus anderen atomaren Constraints folgen, entfernt werden. Die daraus entstehenden Constraints sind zu den ursprünglichen Constraints äquivalent und können diese somit ersetzen. Beispielsweise kann der Constraint $s > t \wedge t > u \wedge u > s$ leicht als unerfüllbar nachgewiesen werden, da mit der Transitivität von $>$ die unerfüllbare Beziehung $s > s$ gefolgert werden. Wäre der dritte atomare Constraint $s > u$ statt $u > s$, so wäre dieser überflüssig, da er wegen der Transitivität bereits in $s > t \ \wedge \ t > u$ enthalten ist und könnte somit entfernt werden.

Konkret funktioniert das Lösungs- und Vereinfachungsverfahren folgendermaßen: es wird eine Menge von atomaren Constraints aus allen Ordnungsconstraints erstellt, zu der anschließend auch Beziehungen zwischen Teiltermen hinzugefügt werden. So wird beispielsweise die Beziehung $f(b) > b$ hinzugefügt, falls der Term $f(b)$ im Constraint auftritt. Dann wird ein transitiver Abschluss gebildet und anschließend wird anhand der konkreten Reduktionsordnung überprüft, ob die einzelnen atomaren Constraints erfüllbar sind. Entstehen in diesem Prozess widersprüchliche atomare Constraints (also beispielsweise $a > b$ und $b < a$) oder wird ein atomarer Constraint als unerfüllbar erkannt, so ist der gesamte Constraint als unerfüllbar nachgewiesen. Wenn ein atomarer Constraint des ursprünglichen Constraints aus anderen atomaren Constraints hergeleitet werden kann (beispielsweise mittels Transitivität), so ist der ursprüngliche atomare Constraint überflüssig und wird entfernt. Dabei wird sicher gestellt, dass keine Zykel auftreten, so dass ein atomarer Constraint sich selbst überflüssig machen könnte.

Eine detaillierte Beschreibung dieses Verfahrens und weitere Ideen für Verbesserungen finden sich auch in [Sch04].

5.7. Hauptschleife

Das System arbeitet mit einer Variante der WALDMEISTER-Loop (siehe [HL02]). Es existieren zwei Mengen, die Menge der *aktiven Klauseln* A und die Menge der *passiven Klauseln* P. Die Menge A entspricht hierbei der oben verwendeten Klauselmenge S, also der Menge, die durch Ziehen von Inferenzen immer mehr angereichert wird. Es wird versucht diese Menge möglichst einfach und klein zu halten. Daher werden möglichst viele – auch sehr aufwändige – Tests durchgeführt, um festzustellen, ob eine Klausel aus A redundant ist. Die Menge P dient der Auswahl der als nächstes zu ziehenden Inferenz. Sie enthält alle Konklusionen der in A ziehbaren Superpositions-Inferenzen in komprimierter Form. Aus diesen wird die als nächstes zu A hinzuzufügende Klausel ausgewählt.

Diese Auswahl geschieht mit Hilfe dreier Warteschlangen. Zunächst gibt es eine bevorzugte Warteschlange. Ist diese nicht leer, so wird die Klausel ausgewählt, die sich am längsten in ihr befindet. Die bevorzugte Warteschlange enthält zu Beginn die Axiome und die negierte Konklusion. Sie kann jedoch auch während des Programmlaufs mit als besonders wichtig erachteten Klauseln gefüllt werden. Neben dieser bevorzugten Warteschlange existiert je eine Warteschlange für Klauseln, die aus einem positiven Literal bestehen, und Klauseln, die aus einem negativen Literal bestehen. Die Klauseln in diesen Warteschlangen besitzen ein nach einer konfigurierbaren Heuristik festgelegtes Gewicht. Es wird jeweils die Klausel mit dem geringsten Gewicht aus der Warteschlange gewählt. Sind beide Warteschlangen nicht leer, wird pro 5 Klauseln, die aus einem positiven Literal bestehen, eine Klausel ausgewählt, die aus einem negativen Literal besteht. Sind alle drei Warteschlangen leer, so muss keine weitere Inferenz mehr gezogen werden. Befindet sich dann die leere Klausel in A, so ist die Behauptung bewiesen, ansonsten gilt die Behauptung nicht.

Für die Berechnung des Gewichtes einer Klausel mit Constraint $s \doteq t \mid \aleph$ wird folgende Heuristik verwendet: Das Gewicht eines Terms ist durch die Anzahl der Funktions- und Variablensymbole in t gegeben. Darauf aufbauend ist das Gewicht eines Literals $s \dot{\simeq} t$, eines atomaren Ordnungsconstraints $s > t$ oder eines atomaren Gleichheitsconstraints $s \doteq t$ die Summe der Gewichte von s und t. Schließlich setzt sich das Gewicht der Klausel $s \dot{\simeq} t \mid \aleph$ zusammen aus:

- 5 mal das Gewicht von $s\sigma \dot{\simeq} t\sigma$, wobei $\sigma := mgu(eqpart(\aleph))$
- 2 mal das Gewicht der atomaren Ordnungsconstraints aus $\aleph$
- -2 mal das Gewicht der atomaren Gleichheitsconstraints aus $\aleph$

Das Gewicht der Klausel $f(g(a,x)) \simeq y \mid x \doteq f(b), y > a$ ist also z. B. $5*(5+1)+2*(1+1)-2*(1+2) = 28$. Diese Heuristik entstand aufgrund weniger Experimente. Es ist wahrscheinlich möglich, durch weitere Experimente bessere Heuristiken zu finden.

Die extrem gute Konfigurierbarkeit der Hauptschleife ergibt sich daraus, dass sie über 4 verschiedene Steckplätze verfügt, an denen frei konfigurierbare Funktionen aufgerufen werden können, die dazu dienen, Inferenzen und Redundanzkriterien umzusetzen (siehe Abbildung 5.1).

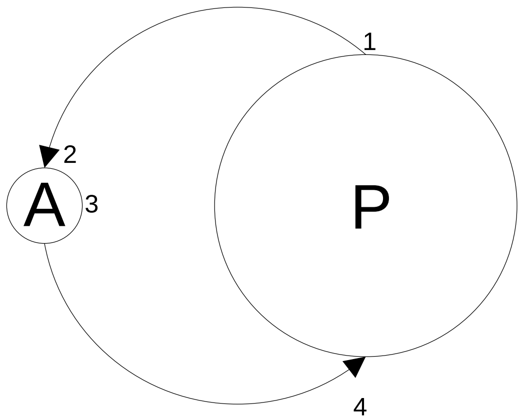


Abbildung 5.1.: Struktur der Hauptschleife

1. Beim Auswählen einer Klausel aus P dienen die konfigurierbaren Funktionen dazu, die ausgewählte Klausel möglichst stark zu vereinfachen und wenn möglich sogar als redundant bezüglich A nachzuweisen. Die hier verwendeten Funktionen entsprechen also hauptsächlich den vorgestellten Simplifikationsregeln und Redundanzkriterien.

2. Nach dem Auswählen einer Klausel aus P wird versucht, die Klauseln aus A mit Hilfe der ausgewählten Klausel zu vereinfachen und evtl. sogar als redundant nachzuweisen.

3. Nach dem Einfügen der neuen Klausel in A werden an dieser Stelle die Konklusionen aller Inferenzen bestimmt, die zwischen der neuen Klausel und den Klauseln aus A ziehbar sind.

4. Die im dritten Schritt erzeugten Klauseln werden vor dem Einfügen in P vereinfacht und es wird versucht, sie als redundant nachzuweisen. Das Vereinfachen erlaubt das Bestimmen sinnvollerer Gewichte und verbessert damit die Heuristik. Wird eine Klausel bereits hier als redundant erkannt, so wird für P weniger Speicherplatz benötigt. Es sind also Rechenzeit gegenüber einer verbesserten Heuristik und Speicherplatzbedarf abzuwägen. Daher werden an dieser Stelle im Allgemeinen weniger aufwändige Tests als im ersten Schritt durchgeführt.

5.8. Umsetzungen

Für die Messungen sollen eine Umsetzung des Inferenzsystems $\mathcal{H}_{\mathcal{U}}$ und verschiedene Konfigurationen einer Umsetzung von $\mathcal{H}_{\mathcal{CU}}$ miteinander verglichen werden. Die Umsetzungen unterscheiden sich im Wesentlichen darin, welche Funktionen von der Hauptschleife aufgerufen werden. Die Umsetzungen von $\mathcal{H}_{\mathcal{CU}}$ verwenden sogar die selben Funktionen und unterscheiden sich nur in der Konfiguration einer dieser Funktionen. Insgesamt werden folgende Umsetzungen verwendet:

Volle Constraints (VC) Eine Umsetzung von $\mathcal{H}_{\mathcal{CU}}$, die sowohl mit Ordnungs- als auch mit Gleichheitsconstraints arbeitet.

Ordnungsconstraints (OC) Eine Umsetzung von $\mathcal{H}_{\mathcal{CU}}$, bei der der Gleichheitsanteil der Constraints sofort in die Klausel gezogen wird, so dass nur noch Ordnungsconstraints verwendet werden.

Referenz (REF) Die Umsetzung von $\mathcal{H}_{\mathcal{U}}$. Diese Umsetzung verwendet keine Constraints, da auch das Inferenzsystem $\mathcal{H}_{\mathcal{U}}$ keine Constraints verwendet.

5.8.1. Umsetzungen von $\mathcal{H}_{\mathcal{CU}}$

Die in den Umsetzungen von $\mathcal{H}_{\mathcal{CU}}$ in der Hauptschleife verwendeten Funktionen (vergleiche Abbildung 5.1) sollen nun genauer vorgestellt werden.

1. Beim Auswählen einer Klausel aus P wird zunächst eine Normalform ihres Constraints gebildet und die Erfüllbarkeit des Constraints mit Hilfe des in Abschnitt 5.6 vorgestellten Verfahrens überprüft. Ist der Constraint nicht unerfüllbar, so werden möglichst viele Variablen abstrahiert.

   Der Constraint der neu entstandenen Klausel wird danach möglichst stark vereinfacht. Dabei werden zunächst Variablen, die der Constraint miteinander identifiziert, zu einer einzigen Variable zusammengefasst. Weiterhin werden Gleichheitsconstraints, die eine in der Klausel sonst nicht vorkommende Variable an einer Term binden, und Ordnungsconstraints, die nur Variablen enthalten, die sonst nicht in der Klausel vorkommen, aufgegeben. Das Aufgeben der Gleichheitsconstraints ist hierbei sicher, das Aufgeben der Ordnungsconstraints entspricht jedoch nur einer Heuristik, die durch weitere Untersuchungen wahrscheinlich verbessert werden kann. Je nach Konfiguration werden bei diesem Vereinfachen des Constraints zusätzlich evtl. alle Ordnungs- bzw. alle Gleichheitsconstraints entfernt. Im Anschluss daran wird versucht, die Klausel zu orientieren.

Dann wird die resultierende Klausel nach Möglichkeit mittels Reduktions-Simplifikation (siehe Abschnitt 5.3.5) simplifiziert. Da sich diese Methode in der Praxis als sehr mächtig erwiesen hat, wird versucht, durch teilweise Aufgabe des Constraints die Bedingungen für diese Simplifikation zu erfüllen. Dazu werden Teile des Gleichheitsanteils des Constraints in die Klausel gezogen, damit evtl. besser ein Match gefunden werden kann, oder Teile des Ordnungsanteils aufgegeben, damit die vierte Bedingung hinsichtlich der Erfüllbarkeit der Constraints erfüllt werden kann. Allerdings wird die vierte Bedingung nur durch einen hinreichenden Test überprüft, ebenso wie die fünfte Bedingung, die relative Reduziertheit (siehe Lemma 26). Nach diesem Schritt wird erneut der Constraint normalisiert und vereinfacht sowie eine Variablenabstraktion durchgeführt.

Im Anschluss wird überprüft, ob die entstandene Klausel eine Tautologie (siehe Abschnitt 5.3.2) ist, indem geprüft wird, ob ihr Constraint unerfüllbar ist oder ob die zwei Seiten einer Gleichung durch den Constraint unifiziert werden.

Daraufhin wird festgestellt, ob die Klausel von einer der bereits in den aktiven Klauseln enthaltenen Klausel subsumiert wird (siehe Abschnitt 5.3.3) und die Klausel gegebenenfalls gelöscht.

Falls es sich bei der neuen Klausel um eine Ungleichung handelt, wird überprüft, ob mit dieser Klausel das Beweisziel bereits erreicht wurde. Dies geschieht, indem versucht wird, mit Hilfe der Inferenzregel *Reflexivitäts-Resolution* die leere Klausel herzuleiten.

2. Nun wird versucht, die bereits in A enthaltenen Klauseln mit der aus P ausgewählten Klausel zu interreduzieren, das heißt zu simplifizieren oder als redundant nachzuweisen. Dazu werden die Subsumption und Reduktions-Simplifikation wie unter 1. beschrieben angewandt, jedoch in umgekehrter Richtung. Das heißt, es wird überprüft, ob die neue Klausel eine der bereits in A enthaltenen Klauseln subsumiert oder simplifiziert.

3. Das Einfügen einer Klausel in die Menge A der aktiven Klauseln führt dann dazu, dass entsprechend der in Abschnitt 5.2 vorgestellten Inferenzregel *Superposition* neue Klauseln erzeugt werden.

4. Jede im dritten Schritt mittels Superposition erzeugte Klausel wird nun in P eingefügt. Dazu wird die Klausel zunächst analog zum ersten Schritt simplifiziert und auf Redundanz überprüft. Im Gegensatz zum ersten Schritt wird jedoch aus Zeitgründen auf den Test verzichtet, ob die Klausel subsumiert wird. Zuletzt wird überprüft, ob eine *Reflexivitäts-Resolution* anwendbar ist und anderenfalls wird die Klausel in eine der Warteschlangen von P eingefügt.

5.8.2. Umsetzung von $\mathcal{H}_{\mathcal{U}}$

Die Umsetzung von $\mathcal{H}_{\mathcal{U}}$ ist den Umsetzungen von $\mathcal{H}_{\mathcal{CU}}$ sehr ähnlich, verwendet aber die stärkeren Redundanzkriterien aus Abschnitt 5.5. Außerdem entfallen alle Schritte, die sich lediglich auf den Constraint beziehen, wie beispielsweise die Variablenabstraktion. Die Schritte der Hauptschleife sehen dann folgendermaßen aus:

1. Beim Auswählen einer Klausel aus P wird zunächst versucht, die Klausel nach Möglichkeit mittels Reduktions-Simplifikation (siehe Abschnitt 5.5.3) zu simplifizieren.

   Im Anschluss wird überprüft, ob die entstandene Klausel eine Tautologie (siehe Abschnitt 5.5.1) ist, indem geprüft wird, ob die zwei Seiten einer Gleichung syntaktisch äquivalent sind.

   Daraufhin wird festgestellt, ob die Klausel von einer der bereits in den aktiven Klauseln enthaltenen Klausel subsumiert wird (siehe Abschnitt 5.5.2) und die Klausel gegebenenfalls gelöscht.

   Falls es sich bei der neuen Klausel um eine Ungleichung handelt, wird überprüft, ob mit dieser Klausel das Beweisziel bereits erreicht wurde. Dies geschieht, indem versucht wird, mit Hilfe der Inferenzregel *Reflexivitäts-Resolution* die leere Klausel herzuleiten.

2. Nun wird versucht, die bereits in A enthaltenen Klauseln mit der aus P ausgewählten Klausel zu interreduzieren, das heißt zu simplifizieren oder als redundant nachzuweisen. Dazu werden die Subsumption und Reduktions-Simplifikation wie unter 1. beschrieben angewandt, jedoch in umgekehrter Richtung. Das heißt, es wird überprüft, ob die neue Klausel eine der bereits in A enthaltenen Klauseln subsumiert oder simplifiziert.

3. Das Einfügen einer Klausel in die Menge A der aktiven Klauseln führt dann dazu, dass entsprechend der in Abschnitt 5.4 vorgestellten Inferenzregel *Superposition* neue Klauseln erzeugt werden.

4. Die im 3. Schritt mittels Superposition erzeugten Klauseln werden nun in P eingefügt. Dabei wird zunächst wie unter 1. festgestellt, ob die Klausel eine Tautologie ist. Anschließend wird überprüft, ob eine *Reflexivitäts-Resolution* anwendbar ist und dann wird die Klausel in eine der Warteschlangen von P eingefügt.

6. Ergebnisse der Vergleichsmessungen

6.1. Versuchsbeschreibung

Für die Messungen wurden 192 Probleme aus der TPTP 2.2.0 [SS99] verwendet. Die Auswahl der Probleme wurde so getroffen, dass sie zwar unterschiedlich schwere Probleme enthält, aber jedes Problem von zumindest einer Umsetzung innerhalb eines Zeitlimits von 15 Minuten gelöst werden konnte. Die Messungen wurden durchgeführt auf Intel Pentium III 1 GHz Maschinen mit 4 GB Arbeitsspeicher.

Dabei konnten 99 dieser Probleme von allen Umsetzungen innerhalb des Zeitlimits gelöst werden. In Tabelle 6.1 ist eingetragen, wie viele der Probleme von den unterschiedlichen Umsetzungen gelöst werden konnten. Die genauen Messergebnisse sind in Anhang A aufgelistet.

Umsetzung	**gelöste Probleme**
OC	117
VC	102
REF	191

Tabelle 6.1.: Anzahl der gelösten Probleme

Um einen objektiven Vergleich der verschiedenen Umsetzungen zu ermöglichen, ist es notwendig, Kriterien zu finden, mit denen sich Aussagen über die Qualität des verwendeten Inferenzsystems und der Heuristik treffen lassen. Die reine Laufzeit ist daher ungeeignet, da sie mehr Aussagen über die Implementierung als über das zu testende Inferenzsystem zulässt. Besser geeignet ist die Zahl der Iterationen der Hauptschleife, die eine ungefähre Abschätzung des Aufwandes ermöglicht. Da es Ziel der Verwendung von Constraints ist, die Erzeugung von Klauseln einzuschränken, wird die Gesamtzahl der erzeugten Klauseln als zusätzliches Vergleichskriterium erfasst. Zudem stehen bei den verschiedenen Umsetzungen unterschiedlich starke Redundanzkriterien zur Verfügung. Diese Kriterien sind entscheidend dafür, welche erzeugten Klauseln weiter betrachtet, das heißt in P eingefügt werden müssen. Daher ist es auch interessant, die Gesamtzahl der in P eingefügten Klauseln zu betrachten.

Es ergeben sich daher die folgenden Vergleichskriterien:

- Hauptschleifeniterationen
- erzeugte Klauseln
- in P eingefügte Klauseln

Beim Vergleich zweier Umsetzungen sollen jeweils nur die Probleme berücksichtigt werden, die innerhalb des Zeitlimits von beiden Umsetzungen gelöst werden konnten. Tabelle 6.2 zeigt, wie viele Probleme jeweils in den Vergleich zweier Konfigurationen eingehen.

Vergleich	Probleme
REF - OC	117
REF - VC	102
OC - VC	99

Tabelle 6.2.: Anzahl der bei den Vergleichen berücksichtigten Probleme

6.2. Auswertung der Messergebnisse

Im Folgenden sollen die Umsetzungen, die Constraints verwenden, also OC und VC, jeweils mit der Referenz REF sowie untereinander verglichen werden. Dazu wird für jedes Vergleichskriterium ein Diagramm erstellt, in dem für jedes von beiden Umsetzungen gelöste Problem der Wert der einen Umsetzung auf der Abszisse und der Wert der zu vergleichenden Umsetzung auf der Ordinate eingetragen wird. Jeder Punkt, der oberhalb der ersten Winkelhalbierenden liegt, repräsentiert daher ein Problem, das in Bezug auf das Vergleichskriterium bei der auf der Abszisse eingetragenen Umsetzung einen niedrigeren und damit besseren Wert erzielt als bei der auf der Ordinate eingetragenen Umsetzung. Umgekehrt repräsentieren Punkte unterhalb der Winkelhalbierenden Probleme, bei denen die Umsetzung auf der Ordinate besser ist. Zur Orientierung ist daher die erste Winkelhalbierende ebenfalls in die Diagramme eingetragen. Die beiden Achsen sind dabei logarithmisch skaliert.

6.2.1. Ordnungsconstraints

Vergleicht man die Umsetzung, die nur Ordnungsconstraints verwendet (OC), mit der Referenz (REF), so fällt auf, dass in allen drei Diagrammen die Mehrzahl der Punkte deutlich oberhalb der Diagonalen liegt (siehe Abbildung 6.1). Das heißt, dass die Verwendung von Ordnungsconstraints in den meisten Fällen weder hinsichtlich der Hauptschleifeniterationen noch hinsichtlich der erzeugten oder der in P eingefügten Klauseln einen Vorteil bringt. Bemerkenswert sind jedoch einige Ausreißer, die deutlich unterhalb der Diagonalen liegen, insbesondere bei der Zahl der erzeugten Klauseln: Ein Punkt befindet sich ganz rechts im Diagramm und repräsentiert das Problem ROB009-1. Hier werden bei der Referenz 19 887 Klauseln in 68 Hauptschleifeniterationen erzeugt, wohingegen mit Ordnungsconstraints nur 428 Klauseln in 23 Hauptschleifeniterationen erzeugt werden. Der nächste Punkt, der recht deutlich unterhalb der Diagonalen liegt, repräsentiert das Problem ROB013-1. Weitere Messungen mit einem höheren Zeitlimit haben die Vermutung erhärtet, dass die Verwendung von Ordnungsconstraints in der Domäne der Robbins Algebra eine Verbesserung bringen kann. Dies könnte daran liegen, dass bei diesen Problemen besonders stark mit Assoziativität und Kommutativität gearbeitet wird und deren Anwendung durch die Verwendung von Ordnungsconstraints eingeschränkt wird. Genauere Untersuchungen mit einem Problem, das nur Assoziativität und Kommutativität (AC) für das Funktionssymbol $+$ und die beiden Konstanten a und b enthält, zeigen jedoch, dass die Erzeugung der durch AC möglichen Permutationen nicht signifikant eingeschränkt wird, sondern der wesentliche Unterschied in der Reihenfolge ihrer Erzeugung liegt. Dies ist dadurch begründet, dass die Heuristik, nach der die als nächstes zu aktivierende Klausel bestimmt wird, zusätzlich die Ordnungs-

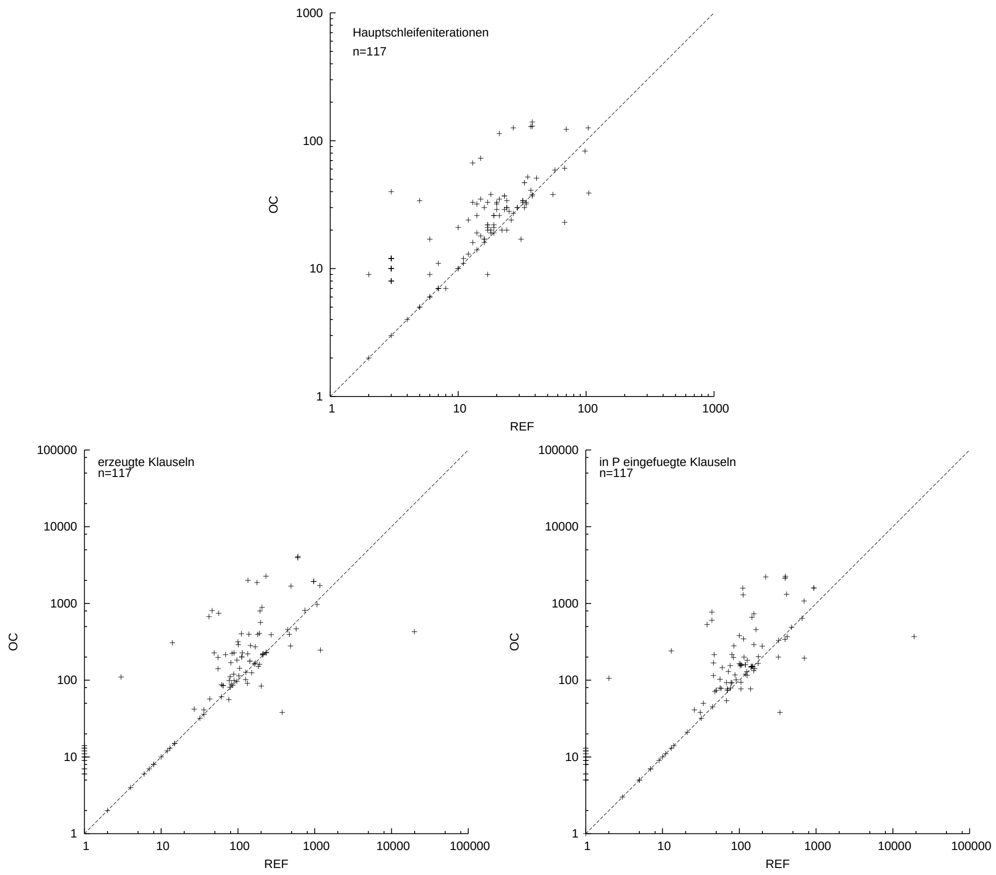


Abbildung 6.1.: Vergleich Referenz mit Ordnungsconstraints

constraints berücksichtigt. Leider macht es daher den Anschein, dass sich bei den zuvor beschrieben Ausreißern die Ordnungsconstraints nicht aufgrund prinzipieller Änderungen positiv auswirken, sondern lediglich die geänderte Heuristik zufällig einen positiven Effekt hat.

Betrachtet man die aktivierten Klauseln, fällt auf, dass viele dieser Klauseln bis auf den Constraint identisch sind. Dies kann auch ein Grund dafür sein, dass die Umsetzung mit Ordnungsconstraints häufig mehr Hauptschleifeniterationen benötigt als die Referenz. Hier muss nach geeigneten Strategien gesucht werden, solche Klauseln zusammenzufassen und gemeinsam zu behandeln. Weiterhin fällt auf, dass gelegentlich Klauseln mit unerfüllbarem Constraint aktiviert werden. Dies liegt daran, dass nur ein hinreichender Erfüllbarkeitstest verwendet wird. Durch Einbeziehung zusätzlicher Informationen über die verwendete Ordnung könnte dieser verbessert werden. Beispielsweise entsteht im zuvor beschriebenen Beispiel folgender Constraint: $+(x_2, x_1) > +(x_1, x_2) \wedge +(x_1, +(x_2, x_3)) > +(x_2, +(x_1, x_3))$. Dieser ist zum Beispiel bei Verwendung einer LPO unerfüllbar, da sich aus dem ersten atomaren Constraint $x_2 > x_1$ und aus dem zweiten atomaren Constraint $x_1 > x_2$ herleiten lässt. Das verwendete hinreichende Verfahren kann diesen Constraint jedoch nicht als unerfüllbar nachweisen, da beide atomaren Constraints für sich mit der LPO erfüllbar sind und es eine Reduktionsordnung gibt, mit der der gesamte Constraint erfüllbar ist. Ein Beispiel für eine solche Reduktionsordnung ist eine Polynomordnung mit folgender

Interpretation:

$$\begin{aligned}\varphi(a) &= 1 \\ \varphi(b) &= 2 \\ \varphi(+)(x,y) &= 6x+2y\end{aligned}$$

Mit dieser Ordnung und der Variablenbelegung σ mit $x_1\sigma = a, x_2\sigma = b, x_3\sigma = a$ wird der Constraint erfüllt.

6.2.2. Volle Constraints

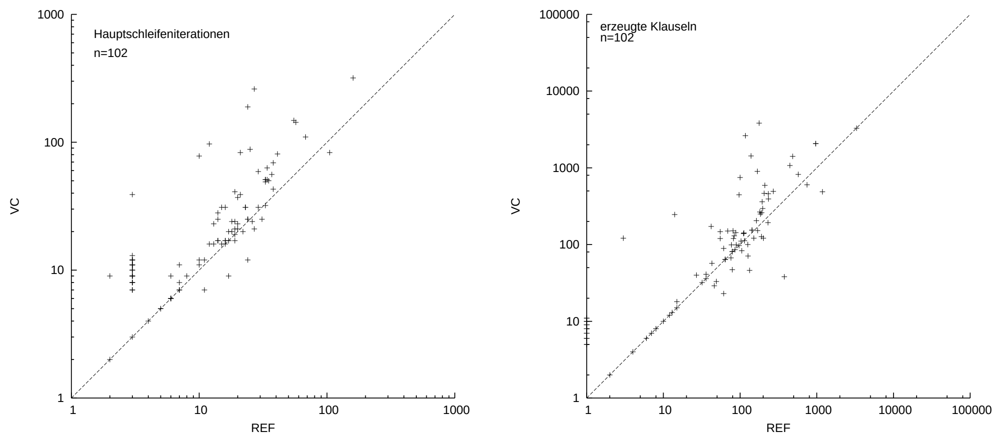


Abbildung 6.2.: Vergleich Referenz mit vollen Constraints

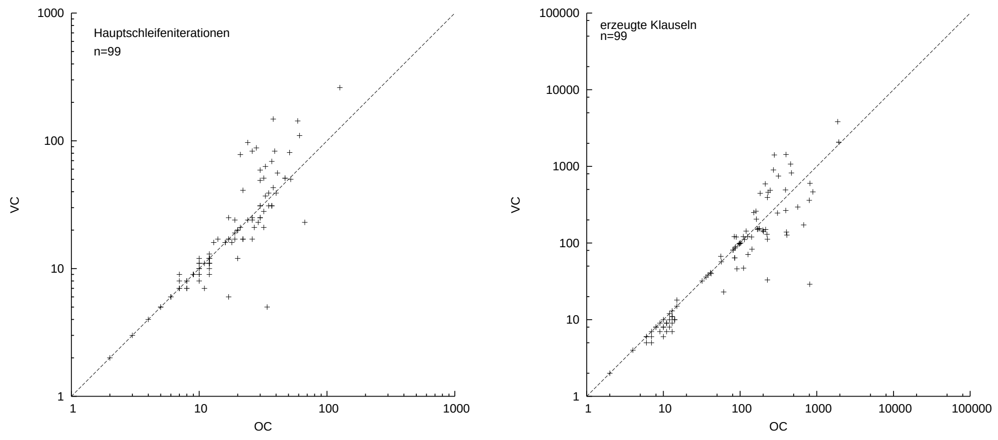


Abbildung 6.3.: Vergleich Ordnungsconstraints mit vollen Constraints

Beim Vergleich der Referenz (REF) mit der Umsetzung, die sowohl Gleichheits- als auch Ordnungsconstraints verwendet (VC), fällt auf, dass sich auch diese Umsetzung wesentlich schlechter als die Referenz verhält (siehe Abbildung 6.2). Es gibt zwar einige Probleme, bei denen der Einsatz von vollen Constraints einen Vorteil bringt, bei den allermeisten Probleme sind sowohl mehr Hauptschleifeniterationen als auch mehr erzeugte Klauseln nötig.

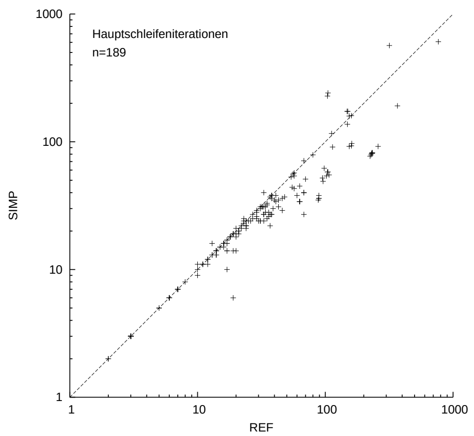
Vergleicht man die Umsetzung VC mit der Umsetzung, die ausschließlich Ordnungsconstraints (OC) verwendet, so fällt ein erkennbarer Unterschied auf, eine eindeutige Tendenz in eine Richtung lässt sich jedoch nicht erkennen (siehe Abbildung 6.3). Lediglich hinsichtlich der Hauptschleifeniterationen kann man feststellen, dass OC bei Problemen, die relativ viele Hauptschleifeniterationen benötigen, ein besseres Verhalten zeigt. An dem deutlichen Unterschied zwischen VC und OC lässt sich erkennen, dass die Gleichheitsconstraints das Verhalten von VC entscheidend beeinflussen. In der Arbeit von Thomas Türk [Tür03] ist dieses Verhalten genauer untersucht und es stellt sich heraus, dass im Wesentlichen die Gleichheitsconstraints das Verhalten von VC bestimmen.

7. Zusammenfassung und Ausblick

In dieser Arbeit wurde der Superpositionskalkül für Hornklauseln mit Constraints vorgestellt und dessen Vollständigkeit bewiesen. Weiterhin wurden für den Unitklausel-Fall Redundanzkriterien entwickelt. Anhand mehrerer Umsetzungen wurden erste Messungen für den Unitklausel-Fall durchgeführt.

Aus der Auswertung der Messergebnisse geht hervor, dass die Verwendung von Constraints nicht nur die erhofften Verbesserungen nicht bringt, sondern sogar zu einer deutlichen Verschlechterung gegenüber der Referenzumsetzung ohne Constraints führt.

Betrachtet man jedoch die Ergebnisse für die Referenz, so fällt auf, dass diese im Vergleich zu den Leistungen anderer Systeme, die beispielsweise mit *Unfailing Completion* arbeiten (siehe [BDP89]), sehr schlecht sind. Einer der Gründe hierfür könnte sein, dass für die Anwendung einer Reduktions-Simplifikation (vgl. Abschnitt 5.3.5) die Bedingung $C \succ D\eta$ erfüllt sein muss. Diese Bedingung verhindert oft die Simplifikation an einer Position, die auf die linke oder rechte Seite des Literals verweist, aus dem C besteht. Solche Simplifikationen sind bei Unfailing Completion jedoch mit kleinen Einschränkungen erlaubt. Entfernt man den Test auf die störende Bedingung $C \succ D\eta$ aus der Referenzumsetzung, so zeigt sich für die entstehende Umsetzung SIMP (siehe Abbildung 7.1) eine deutliche Verbesserung. Ähnliche Verbesserungen zeigen sich auch für die anderen Umsetzungen.



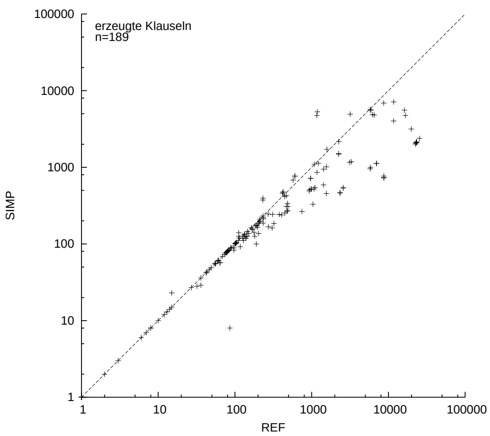


Abbildung 7.1.: Vergleich Referenz und Referenz mit verstärkter Simplifikation

Wie man sieht, ergibt sich hier ein interessantes Gebiet für weitere Verbesserungen der Leistungsfähigkeit des Superpositionskalküls. Dazu müsste versucht werden, die Vollständigkeit des Verfahrens mit einem erweiterten Redundanzbegriff, der solche Simplifikationsschritte zulässt, zu zeigen. Dass dies zumindest für den Unitklausel-Fall möglich ist, legt die Vollständigkeit der Unfailing Completion nahe.

Auch die verwendeten Heuristiken bieten einen Ansatzpunkt für weitere Verbesserungen. Insbesondere sind hier die Heuristik für die als nächstes zu aktivierende Klausel und die Heuristiken für die Aufgabe von Constraints zu untersuchen.

Der Superpositionskalkül mit Constraints wurde hier für allgemeine Hornklauseln entwickelt, jedoch nur für Unitklauseln umgesetzt. Es gibt Grund zu der Annahme, dass die Verwendung von Constraints für allgemeine Hornklauseln mehr Vorteile bringt als für Unitklauseln. Denn bei allgemeinen Hornklauseln können in der Regel weniger Simplifikationsschritte durchgeführt werden als bei Unitklauseln. Daher ist eine Einschränkung der Erzeugung von Klauseln unter Schwächung der Simplifikation wahrscheinlich gewinnbringender. Noch stärker gelten diese Argumente für allgemeine Klauseln. Für weitere Arbeiten wäre es daher lohnenswert, für allgemeine Hornklauseln Redundanzkriterien zu entwickeln oder Untersuchungen für allgemeine Klauseln anzustellen.

Leider zeigen die Messergebnisse nicht den erhofften Erfolg beim Einsatz von Constraints. In der hier vorgestellten Form ist der Einsatz von Constraints im Unitklausel-Fall somit nicht sinnvoll. Es ergibt sich jedoch sowohl für Unitklauseln als auch für allgemeine Hornklauseln großes Potenzial für Verbesserungen. Weitere Arbeiten im theoretischen und praktischen Bereich müssen zeigen, ob dieses Potenzial ausreicht, um Constraints erfolgreich einsetzen zu können.

A. Messergebnisse

Wie in Abschnitt 6 beschrieben wurden 192 Probleme aus der TPTP 2.2.0 [SS99] mit einem Zeitlimit von 15 Minuten und den in Abschnitt 5.8 beschriebenen Umsetzungen VC, OC und REF gemessen. Ausgewertet wurden die Vergleichskriterien

- Hauptschleifeniterationen (HI)
- erzeugte Klauseln (EK)
- in P eingefügte Klauseln (PK).

Bei den folgenden Messergebnissen deutet ein > vor einem Wert in der Tabelle an, dass die Messung wegen des Zeitlimits abgebrochen wurde. In diesem Fall sind die bis zum Abbruch erfassten Werte eingetragen, die tatsächlichen Werte liegen daher über dem angegebenen Wert.

	REF			**OC**			**VC**		
	HI	**EK**	**PK**	**HI**	**EK**	**PK**	**HI**	**EK**	**PK**
BOO001-1	12	73	55	>119	>9450	>2798	>111	>4834	>1822
BOO002-2	107	16651	14457	>104	>10124	>1631	>111	>4903	>1903
BOO003-2	21	135	111	114	2005	1582	>234	>4166	>2689
BOO003-4	15	56	44	73	746	605	>245	>3717	>2603
BOO004-2	36	405	292	>230	>5244	>3428	>233	>3898	>3026
BOO004-4	27	177	112	126	1872	1293	261	3823	2737
BOO005-2	30	273	205	>242	>5726	>3687	>216	>3369	>2300
BOO005-4	22	103	73	>270	>5505	>3139	>204	>3091	>2127
BOO006-2	20	129	107	>230	>6057	>3400	>234	>4224	>2766
BOO006-4	16	60	48	>248	>4955	>2882	>231	>3432	>2490
BOO009-2	33	305	223	>235	>5260	>3436	>234	>3900	>3028
BOO009-4	24	117	81	>234	>4553	>2883	189	2623	1886
BOO010-2	39	477	339	>226	>5748	>3743	>235	>4168	>2691
BOO010-4	29	203	125	>260	>4434	>2748	>238	>3546	>2480
BOO011-2	16	97	78	16	97	92	16	97	96
BOO011-4	11	36	31	12	41	38	12	41	38
BOO012-2	161	16202	9881	>227	>5854	>3795	>235	>4185	>2695
BOO013-2	70	491	413	123	1684	1317	>235	>3916	>2697
BOO013-4	80	1082	452	>337	>4591	>3211	>231	>2976	>2225
BOO016-2	43	487	356	>234	>5396	>3528	>242	>3782	>2969
BOO017-2	37	322	251	>250	>5641	>3821	>238	>3983	>2638
BOO018-4	13	43	34	16	57	50	16	57	50
BOO021-1	24	133	105	20	91	77	12	46	44
BOO024-1	112	1581	1203	>136	>6263	>3741	>183	>2977	>2342
BOO025-1	149	2255	1691	>135	>6190	>3686	>183	>2977	>2342
BOO026-1	364	7826	6611	>172	>2602	>2506	>165	>2120	>2049
COL001-1	40	1227	1180	>110	>4901	>2733	>200	>2192	>1668
COL001-2	29	469	440	>109	>5945	>3773	>130	>3766	>3090
COL004-3	3	14	13	40	307	241	39	246	190
COL044-6	31	2251	2231	>65	>3798	>3611	>64	>2853	>2772
COL044-7	31	2251	2231	>67	>4021	>3830	>64	>2853	>2772

	REF			OC			VC		
	HI	EK	PK	HI	EK	PK	HI	EK	PK
COL050-1	4	4	3	4	4	3	4	4	3
COL051-1	5	7	5	5	7	5	5	7	5
COL052-1	10	15	14	10	15	14	11	18	17
COL053-1	3	2	1	3	2	1	3	2	1
COL056-1	7	8	7	7	8	7	8	8	7
COL057-1	13	232	221	33	2268	2217	>43	>2962	>2945
COL058-1	7	61	58	11	87	78	11	89	78
COL058-2	6	49	47	17	227	215	6	33	27
COL058-3	5	46	44	34	809	771	5	29	23
COL059-1	35	197	165	52	564	458	50	295	282
COL060-2	3	1	1	8	7	6	7	5	4
COL060-3	3	1	1	8	6	5	8	6	5
COL061-2	3	1	1	8	7	6	8	6	5
COL061-3	3	1	1	8	6	5	7	5	4
COL062-2	3	1	1	10	10	9	9	8	7
COL062-3	3	1	1	10	9	8	10	9	8
COL063-2	3	1	1	10	11	10	9	7	6
COL063-3	3	1	1	10	10	9	8	6	5
COL063-4	3	1	1	10	11	10	11	9	8
COL063-5	3	1	1	10	10	9	10	8	7
COL063-6	3	1	1	10	9	8	9	7	6
COL064-10	3	1	1	12	13	12	12	11	10
COL064-2	3	1	1	12	13	12	9	7	6
COL064-3	3	1	1	12	12	11	10	8	7
COL064-4	3	1	1	12	13	12	13	11	10
COL064-5	3	1	1	12	12	11	12	10	9
COL064-6	3	1	1	12	11	10	11	9	8
COL064-7	3	1	1	12	14	13	12	10	9
COL064-8	3	1	1	12	13	12	11	9	8
COL064-9	3	1	1	12	14	13	11	10	9
COL066-2	33	970	935	47	1937	1594	51	2069	1998
COL066-3	33	970	935	47	1937	1594	51	2069	1998
COL075-2	14	82	79	>142	>6999	>2534	>138	>3660	>1340
GRP001-2	19	88	56	22	120	79	41	143	105
GRP001-4	15	79	48	18	111	71	16	47	35
GRP002-2	317	3176	2170	>467	>3994	>3014	>463	>2512	>2144
GRP010-4	14	36	32	14	36	32	17	36	33
GRP011-4	21	97	56	26	183	103	83	445	311
GRP012-4	19	190	103	26	407	154	24	127	64
GRP014-1	114	11718	10297	>34	>4871	>4073	>26	>1347	>1328
GRP022-2	10	32	21	10	32	21	12	32	25
GRP023-2	7	13	10	7	13	10	7	13	10
GRP115-1	14	111	100	19	402	381	17	139	127
GRP116-1	20	194	146	33	798	660	37	361	295
GRP117-1	12	84	80	13	224	216	16	130	119
GRP118-1	21	206	155	35	890	733	39	465	370
GRP136-1	6	8	7	6	8	7	6	8	7
GRP137-1	19	86	77	19	86	79	19	86	79
GRP138-1	150	5929	2986	>216	>6326	>3448	>233	>3626	>2323
GRP139-1	17	77	68	20	99	93	20	99	93
GRP140-1	148	5792	2911	>219	>6813	>3690	>228	>3941	>2561
GRP141-1	23	112	104	37	201	162	31	141	129

	REF			OC			VC		
	HI	**EK**	**PK**	**HI**	**EK**	**PK**	**HI**	**EK**	**PK**
GRP142-1	19	105	88	26	143	117	17	83	76
GRP143-1	15	69	60	35	215	146	31	150	131
GRP144-1	16	81	72	30	169	130	31	151	132
GRP145-1	31	201	140	17	84	77	25	121	109
GRP146-1	18	89	80	20	99	93	20	99	93
GRP147-1	160	6539	3324	>216	>6326	>3446	>231	>3616	>2311
GRP148-1	154	6208	3165	>219	>6813	>3690	>229	>3901	>2525
GRP149-1	23	112	104	37	201	162	31	141	129
GRP150-1	32	212	143	33	218	148	>212	>3827	>2457
GRP151-1	5	10	9	5	10	9	5	10	9
GRP152-1	32	212	143	34	220	150	>212	>3827	>2457
GRP153-1	16	79	70	17	81	74	17	81	74
GRP155-1	259	25488	18962	>203	>6906	>3716	>225	>3909	>2511
GRP158-1	224	22528	16743	>193	>6426	>3484	>222	>3641	>2336
GRP159-1	228	22799	16996	>193	>6426	>3484	>222	>3641	>2336
GRP160-1	7	15	13	7	15	13	7	15	13
GRP161-1	6	12	11	6	12	11	6	12	11
GRP162-1	41	271	199	51	391	278	81	494	370
GRP163-1	38	234	176	37	227	166	69	393	288
GRP165-1	55	484	323	38	279	200	148	1411	1061
GRP165-2	22	126	105	20	102	94	20	100	92
GRP166-1	104	1168	698	126	1706	1078	>287	>3297	>2439
GRP166-2	105	1186	704	39	247	194	83	487	377
GRP166-3	98	1071	655	83	970	641	>290	>3334	>2471
GRP166-4	26	151	126	24	125	116	24	121	112
GRP171-1	154	3127	1596	>254	>5682	>3275	>322	>3264	>2500
GRP171-2	68	573	394	61	467	342	110	822	607
GRP172-1	57	446	325	59	455	330	143	1071	872
GRP172-2	160	3299	1677	>296	>5992	>3173	318	3285	2422
GRP173-1	33	184	154	30	151	133	49	250	216
GRP174-1	34	191	159	32	161	142	51	260	225
GRP176-1	14	55	46	26	141	115	25	120	108
GRP176-2	14	55	46	32	198	168	28	147	133
GRP182-1	18	101	84	38	290	198	>211	>3825	>2455
GRP182-2	24	146	115	34	283	200	>204	>4235	>2258
GRP182-3	17	89	76	33	227	155	>211	>3825	>2455
GRP182-4	23	134	107	29	220	157	>204	>4235	>2258
GRP184-4	33	231	176	>272	>6116	>3835	32	193	170
GRP185-1	102	5847	4466	>227	>6128	>3351	>264	>3909	>2528
GRP185-2	96	5792	4424	>228	>6095	>3350	>263	>4327	>2339
GRP185-3	35	234	160	>213	>6485	>3479	>234	>3880	>2499
GRP185-4	31	232	164	>212	>6581	>3524	>228	>4553	>2420
GRP186-3	29	169	125	30	168	129	31	152	133
GRP186-4	20	114	100	32	227	164	21	112	102
GRP188-1	32	212	143	34	220	150	>212	>3827	>2457
GRP188-2	29	210	147	30	213	152	59	592	415
GRP189-1	16	79	70	17	81	74	17	81	74
GRP189-2	19	103	91	21	114	101	21	111	101
GRP190-1	232	23165	17209	>203	>6906	>3717	>227	>3964	>2563
GRP190-2	230	23053	17103	>202	>6867	>3698	>226	>3914	>2517
GRP191-1	234	23357	17399	>203	>6940	>3732	>218	>3879	>2505
GRP191-2	233	23301	17346	>204	>6979	>3751	>218	>3879	>2505

	REF			OC			VC		
	HI	EK	PK	HI	EK	PK	HI	EK	PK
GRP192-1	25	168	127	28	272	182	88	900	741
GRP195-1	17	1040	1020	>45	>4508	>4441	>40	>3355	>3242
LAT008-1	37	746	478	41	814	490	56	601	430
LAT014-1	11	61	45	11	61	45	7	23	22
LCL110-2	36	948	702	>136	>9617	>6152	>205	>4921	>4631
LCL112-2	38	995	727	>135	>9558	>6115	>203	>4801	>4514
LCL113-2	48	1556	1121	>131	>9115	>5774	>198	>4657	>4374
LCL114-2	41	1172	835	>134	>9496	>6080	>203	>4801	>4514
LCL115-2	46	1428	1024	>137	>9797	>6194	>206	>4957	>4665
LCL116-2	57	2251	1636	>136	>9617	>6152	>205	>4921	>4631
LCL132-1	8	76	68	7	56	54	9	67	64
LCL133-1	20	181	155	29	394	291	23	266	210
LCL134-1	10	100	85	21	316	280	78	748	667
LCL135-1	12	139	114	24	397	345	97	1429	1312
LCL139-1	35	924	688	>136	>9617	>6152	>204	>4904	>4616
LCL140-1	38	995	727	>131	>9115	>5774	>198	>4657	>4374
LCL141-1	46	1428	1024	>131	>9115	>5774	>198	>4657	>4374
LCL153-1	56	423	351	>320	>5226	>3013	>298	>2940	>2206
LCL154-1	95	1557	1061	>334	>5479	>3153	>304	>3047	>2284
LCL155-1	60	447	359	>320	>5226	>3013	>298	>2940	>2206
LCL156-1	54	416	347	>334	>5479	>3153	>301	>3008	>2259
LCL157-1	43	310	270	>320	>5226	>3013	>298	>2940	>2206
LCL158-1	57	427	355	>323	>5236	>3021	>301	>2946	>2212
LCL161-1	34	233	177	33	232	202	63	459	392
LCL164-1	63	1106	723	>225	>6974	>4993	>298	>4558	>3213
LDA001-1	38	163	150	38	163	150	43	205	186
LDA002-1	767	8673	8116	>382	>2872	>2639	>364	>2000	>1941
LDA007-3	27	127	120	27	127	120	21	71	68
RNG007-4	18	82	70	19	89	77	24	120	108
RNG008-3	105	7001	5487	>242	>7279	>4314	>246	>4046	>2915
RNG008-4	104	6993	5483	>242	>7280	>4320	>245	>4075	>2955
RNG008-7	367	11691	8834	>180	>8715	>4448	>209	>4726	>2320
RNG011-5	13	42	38	67	675	531	23	172	159
RNG012-6	89	8717	6700	>202	>7430	>3833	>164	>4209	>3021
RNG013-6	37	601	395	129	3960	2143	>164	>4209	>3021
RNG014-6	88	8716	6699	>202	>7452	>3851	>164	>4209	>3021
RNG015-6	89	8717	6700	>187	>6995	>3581	>162	>4009	>2882
RNG016-6	38	602	396	130	3961	2144	>165	>4210	>3022
RNG017-6	38	605	399	140	4107	2263	>167	>4156	>2989
RNG018-6	89	8719	6702	>188	>6896	>3529	>164	>3952	>2849
RNG019-6	63	2348	1645	>136	>3081	>1562	>164	>2202	>1586
RNG019-7	68	2576	1819	>195	>3283	>2267	>190	>1849	>1436
RNG020-6	63	2348	1645	>136	>3081	>1562	>170	>2262	>1651
RNG020-7	68	2576	1819	>193	>3223	>2241	>188	>1821	>1424
RNG021-6	63	2348	1645	>136	>3081	>1562	>166	>2131	>1531
RNG021-7	68	2576	1819	>197	>3320	>2282	>188	>1825	>1425
RNG023-6	17	64	50	22	85	74	17	64	59
RNG023-7	24	143	120	30	177	159	25	154	146
RNG024-6	17	64	50	22	85	74	17	64	59
RNG024-7	24	143	120	30	177	159	25	154	146
ROB002-1	17	467	420	21	396	369	>80	>2862	>2668
ROB009-1	68	19887	18924	23	428	370	>86	>2000	>1900

	REF			**OC**			**VC**		
	HI	**EK**	**PK**	**HI**	**EK**	**PK**	**HI**	**EK**	**PK**
ROB010-1	6	27	26	9	42	41	9	40	39
ROB013-1	17	377	338	9	38	38	9	38	38
SYN080-1	2	6	5	2	6	5	2	6	5
SYN083-1	2	3	2	9	110	106	9	121	117

Literaturverzeichnis

[Ave95] Avenhaus, Jürgen: *Reduktionssysteme*. Springer, 1995

[BDP89] Bachmair, Leo ; Dershowitz, Nachum ; Plaisted, David A.: Completion Without Failure. In: Aït-Kaci, H. (Hrsg.) ; Nivat, M. (Hrsg.): *Resolution of Equations in Algebraic Structures* Bd. 2: Rewriting Techniques. New York : Academic Press, 1989, S. 1–30

[BGLS95] Bachmair, L. ; Ganzinger, H. ; Lynch, Chr. ; Snyder, W.: Basic Paramodulation. In: *Information and Computation* 121 (1995), Nr. 2, S. 172–192

[GN01] Ganzinger, H. ; Nieuwenhuis, R.: Constraints and Theorem Proving. In: Comon, H. (Hrsg.) ; Marché, C. (Hrsg.) ; R., Treinen (Hrsg.): *Contraints in Computational Logics, International Summer School, September 5–8, 1999, Gif-sur-Yvette, France, Edition* Bd. 2002. Springer-Verlag, 2001, S. 159–201

[HL02] Hillenbrand, Th. ; Löchner, B.: The Next Waldmeister Loop. In: Voronkov, A. (Hrsg.): *Proceedings of the 18th International Conference on Automated Deduction*, Springer-Verlag, 2002 (LNAI), S. 486–500

[KKR90] Kirchner, Claude ; Kirchner, Hélène ; Rusinowitch, M.: Deduction with symbolic constraints / INRIA. 1990 (1358). – Forschungsbericht

[Nie93] Nieuwenhuis, Robert: Simple LPO Constraint Solving Methods. In: *Information Processing Letters* 47 (1993), Nr. 2, S. 65–69

[Sch04] Schmidt, Christian. *Integration eines constraint-basierten Redundanzkriteriums in den Theorembeweiser* Waldmeister. Projektarbeit, Technische Universität Kaiserslautern, Fachbereich Informatik. 2004

[SS99] Suttner, Christian B. ; Sutcliffe, Geoff: The TPTP Problem Library. 1999. – Forschungsbericht

[Tür03] Türk, Thomas. *Constraintbasierte Vervollständigungstechniken: Gleichheitsconstraints*. Projektarbeit, Technische Universität Kaiserslautern, Fachbereich Informatik. Dezember 2003

Index