Seminar
Entscheidungsverfahren für logische Theorien

Die kongruente Hülle

Sebastian Bochra

01.03.2005

Inhaltsverzeichnis

1 Worum geht es bei der kongruenten Hülle?

Die kongruente Hülle hilft uns bei der Feststellung von Implikationen von Grundgleichungen (prädikatenlogische Terme über einer bestimmten Signatur ohne Variablen) die aus mehreren anderen Grundgleichungen folgen. Betrachten wir zum Beispiel die folgende Situation. Es sind mehrere Funktionen gegeben und es soll überprüft werden ob eine Gleichung aus einer anderen folgt. Die Funktionen lauten wie folgt:

$$f(f(a,b),b) = a$$
$$f(a,b) = a$$

Nach kurzer Betrachtung stellen wir fest, das wir das Ergebnis von $f(a,b) = a$ in die erste Formel einsetzen können und somit wiederum die zweite entsteht. Damit ist die Implikation gefunden. Für ein so kleines Gleichungssystem braucht man keinen ausgefallenen Algorithmus bemühen. Besteht das System allerdings aus mehreren hundert Funktionen mit vielen verschiedenen Variablen stößt man schnell an die Grenzen der Überschaubarkeit. Es wird ein Werkzeug benötigt, welches bei der Bestimmung der Implikationen und Gleichheiten von in Bezug gesetzter Gleichungen helfen kann, denn beinahe alle Beweise werden darüber geführt.

Deshalb bemüht man automatisch ablaufende Verifikationsmethoden. Die Idee der mechanischen Verifikation wurde bereits 1954 erdacht, allerdings wurde damals kein konkreter Algorithmus mit angegeben. Erst 1976/77 entwarfen mehrere Wissenschaftler verschiedene Lösungsansätze die im Kern die gleiche Herangehensweise verwenden. Sie benutzen die Methode der kongruenten Hülle, welche über die Relation auf einen Graphen berechnet wird.

2 Kurzer Einschub: Äquivalenzrelationen

Um die kongruente Hülle zu berechnen werden wir eine kongruente Äquivalenzrelation benötigen. Hier sein noch mal die Eigenschaften derselbigen aufgezählt. Sei R eine Äquivalenzrelation über der Menge M. R ist:

1. reflexiv. $\forall x \in M$ gilt: $R(x,x)$
2. symmetrisch. $\forall x,y \in M$ gilt: $R(x,y) \Leftrightarrow R(y,x)$
3. transitiv. $\forall x,y,z \in M$ gilt: $R(x,y) \wedge R(y,z) \Rightarrow R(x,z)$

Äquivalenzrelationen bilden Äquivalenzklassen, in welchen alle Elemente der Relation eingeteilt werden. Eine solche Klasse enthält alle Elemente die zu einander äquivalent sind. Äquivalenzklassen einer Relation sind entweder identisch oder disjunkt. Monotonie bedeutet:

- $\forall x,y : x = y \Rightarrow f(x) = f(y)$

Bei zweistelligen Funktionen bedeutet das:

- $\forall x, y, z : x = y, \Rightarrow g(x, z) = g(y, z)$
- $\forall x, y, z : x = y, \Rightarrow g(z, x) = g(z, y)$

Eine monotone Äquivalenzrelation ist eine Kongruenz.

3 Wie funktioniert die kongruente Hülle?

Beginnen wir am besten mit einem Beispiel. Dazu betrachten wir eine weitere Menge aus Gleichungen:

$$f(a, b) = a \tag{1}$$

$$f(f(a, b), b) = c \tag{2}$$

$$g(a) \neq g(c) \tag{3}$$

Nach einigen Umformungen können wir folgern das diese Menge inkonsistent ist. Dazu können wir die Erkenntnis benutzen, dass Gleichheit eine kongruente Äquivalenzrelation ist.

Aus Gleichung 1 folgt:

$$f(f(a, b), b) = f(a, b) \tag{4}$$

Mit Hilfe von Gleichung 4, Symmetrie auf 4, 2 und Transitivität folgt:

$$f(a, b) = c \tag{5}$$

Gleichung 5, 1, Symmetrie auf 1 und Transitivität führt zu:

$$a = c \tag{6}$$

Und mit Hilfe der Kongruenz auf Gleichung 6 angewandt folgt:

$$g(a) = g(c) \tag{7}$$

Dieses widerspricht jedoch Gleichung 3.

Um diesen Sachverhalt mit einem Algorithmus erstens darstellen und zweitens nachvollziehen zu können, wird eine spezielle Datenstruktur benötigt.

3.1 Directed Acyclic Graph

Bei dieser Datenstruktur handelt es sich um einen DAG.(Directed Acyclic Graph). Einen *gerichteten*, *geordneten*, *azyklischen multigraphen*, dessen Knoten benannt sind. Die Knoten des DAG repräsentieren Grundterme der gegebenen Grundgleichungen. Azyklisch bedeutet das es keinen Knoten im Graphen gibt, für den man einen Weg finden könnte welcher den Knoten mehr als ein mal durchquert. Geordnet bedeutet, dass die Kanten die einen Knoten verlassen eine Reihenfolge besitzen. Gerichtet heißt das sie nur in eine Richtung einen Knoten verlassen. Hinzu kommt noch das in einem Multigraphen von einem Knoten mehrere Kanten zu einem anderen Knoten führen können.

3.1.1 Der Graph

Gibt es eine Kante von Knoten u zu Knoten v, so nennen wir u einen Elternknoten von v und v ein Kind von u. $\lambda(u)$ bezeichnet den Namen des Knotens und $\delta(u)$ benennt die Anzahl der Kanten die u verlassen. Mit $u[i]$ wird das $i-te$ Kind von u bezeichnet. Dabei sind die Kinder nach der Reihenfolge des Knoten u geordnet. Die Bezeichnung $Kinder[u]$ kennzeichnet die Sequenz $u[1], \ldots, u[\delta(u)]$

Der Knoten u stellt den Term $f(t_1, \ldots, t_n)$ dar, indem $\lambda(u) = f$ ist und $Kinderu$ ist eine Sequenz von Knoten $v_1, \ldots, v_n$, wobei jedes v_i ein t_i repräsentiert.

3.1.2 Die Kongruenz

Nun wird eine Äquivalenzrelation R definiert, über die Knoten eines Term-Graphen, wie er im vorherigen Abschnitt beschrieben wurde. Zwei Knoten u, v des Graphen G sind unter R *kongruent* wenn $\lambda(u) = \lambda(v)$, $\delta(u) = \delta(v)$, sowie $(\forall i : 1 \leq i \leq \delta(u) | R(u[i], v[i]))$

Dies bedeutet das die Knoten u und v den gleichen Namen haben, die gleiche Anzahl Kinder haben und alle Kinder paarweise zueinander kongruent sind.

Die Relation R ist unter *Kongruenz geschlossen*, wenn zwei kongruente Knoten im Graphen zu einander kongruent und äquivalent sind. Die kongruente Hülle einer Relation R auf einem Term-Graphen ist die kleinste unter *Kongruenz geschlossene* Äquivalenzrelation.

3.1.3 Ein ausführliches Beispiel

Wir bemühen für eine ausführliche Darstellung des Sachverhaltes noch einmal das Beispiel aus dem Vorherigen Abschnitt. Gegeben sind die folgenden 3 Grundgleichungen:

- $f(a, b) = a$

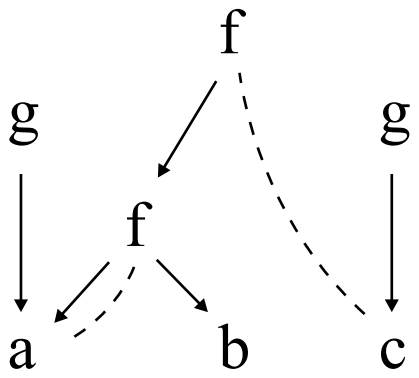

Abbildung 1: Der Ausgangsgraph

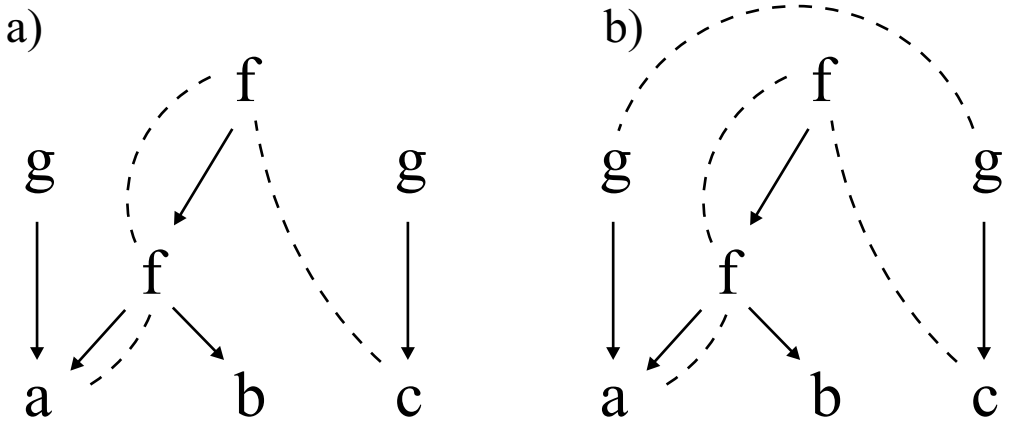

Abbildung 2: Der bearbeitete Graph

- $f(f(a,b),b) = c$
- $g(a) \neq g(c)$

Die Grafik 1 gibt den Term-Graphen wieder.

Wie besprochen bezeichnen die Knoten die Funktionssymbole. Hier f, a, b, c. Der Pfad entlang der Kanten vom obersten Knoten mit dem Namen f bezeichnet den Term $f(f(a,b)b)$. Die gestrichelten Linien folgen den Äquivalenzen aus den gegebenen Termen. So gibt es zum Beispiel vom unteren Knoten mit Namen f, der ja den Term $f(a,b)$ darstellt eine Äquivalenz zum Knoten a, was der Gleichung $f(a,b) = a$ entspricht.

Um die kongruente Hülle zu erhalten müssen wir nun Schritt für Schritt die Knoten des Graphen auf Kongruenz untersuchen und diese in Äquivalenzklassen einordnen, sowie Äquivalenzklassen miteinander vereinen, so-

bald diese identisch sind. Sollten hierbei Äquivalenzklassen miteinander vereint werden, welche den gegebenen Gleichungen nach nicht äquivalent sein dürfen, so ist das Gleichungssystem nicht erfüllbar. In Bild 2 wird dieser Vorgang vorgeführt. In a) wurde die Kongruenz der Knoten $f(f(a,b),b)$ und $f(a,b)$ festgestellt. Sie wurden äquivalent gemacht(gestrichelte Linie). In b) wurde die Kongruenz von $g(a)$ und $g(c)$ festgestellt. Sie wurden ebenfalls äquivalent gemacht.(Knoten a und b sind bereits innerhalb einer Äquivalenzklasse, das heißt, sie sind bereits kongruent.) Und auch an dem Graphen erhalten wir das gleiche Ergebnis, wie bei der Herleitung im Kapitel zuvor: $g(a)$ und $g(c)$ sind äquivalent, obwohl Gleichung Nr. 3 das Gegenteil fordert.

3.1.4 Die Berechnung der kongruenten Hülle

Um diesen Vorgang als Algorithmus wiedergeben zu können benötigen wir einige kleine Erweiterungen.

- Die Funktion *union(u,v)* vereint die Äquivalenzklassen der Knoten u und v
- Die Funktion *find(u)* gibt einen Eindeutigen Repräsentanten der Äquivalenzklasse vom Knoten u zurück.

Es sei: R eine Relation über den Knoten des Graphen G. R sei unter Kongruenz geschlossen und u und v Knoten von G. Der folgende Algorithmus erstellt die kongruente Hülle der Relation R.

- *vereine(u,v)*

1. Wenn *find(u)* = *find(v)* return
2. Sei P_u die Menge aller Eltern aller zu u äquivalenter Knoten, P_v analog zu v
3. *union(u,v)*
4. Für alle Paare (x,y) so das $x \in P_u \wedge y \in P_v$: wenn *find(u)*≠*find(v)* aber *kongruent(x,y)* = true, dann *vereine(x,y)*

- *kongruent(x,y)*

1. wenn $\delta(u) \neq \delta(v)$, dann return false
2. for $i = 1, i \leq \delta(u), i++$: wenn *find(u[i])*≠*find(v[i])*, dann return false
3. return true

4 Die Abstrakte kongruente Hülle

Wie bereits erwähnt, wurden in den siebziger Jahren mehrere Algorithmen zur Programmverifikation erdacht die auf der kongruenten Hülle basieren. Besonders hervorzuheben sind hier die Algorithmen von Nelson und Oppen, Shostak sowie Downey-Sethi-Tarjan. Die abstrakte kongruente Hülle ist eine Verallgemeinerung dieser Algorithmen und lässt sich auf eine kleine Menge von Regeln zurückführen, mit welchen eben diese Algorithmen wiedergegeben werden können.

4.1 Voraussetzungen

Zuerst werden einige Definitionen notwendig. Als erstes fassen wir Funktionssymbole und Konstanten zu einer *Signatur* zusammen: $\Sigma = \cup_n \Sigma_n$. Der Index n der Menge Σ_n bezeichnet die *Arität* eines Funktionssymbols $f \in \Sigma_n$. Beispiel: Wenn $n = 3$ ist, sieht die Funktion so aus: $f(t_1, t_2, t_3)$. Bei $n = 1$ ist es $f(t_1)$. Bei Funktionen mit der Arität 0 handelt es sich um *Konstante*. Die Menge von Grundtermen $\mathrm{T}(\Sigma)$ über Σ ist die kleinste Menge die Σ_0 enthält und es immer gilt das $f(t_1, \ldots, t_n) \in \mathrm{T}(\Sigma)$ wann immer $f \in \Sigma_n$.

Mit den Bezeichnern $f, g, \ldots$ benennen wir Funktionssymbole, mit $s, t, \ldots$ Terme, mit c Konstanten. $t[s]$ bedeutet den Term t, welcher den Term s als Unterterm enthält. Dem zu Folge bedeutet $t[u]$ das das Vorkommen von s in t durch u ersetzt wurde.

Eine *Gleichung* ist ein Paar von Termen, geschrieben als $s \approx t$. Eine *Ersetzung*, oder eine *einzel-Schritt-Umschreibung* $\rightarrow_E$ über einer Menge von Grund-Gleichungen E wird definiert als

$$t[l] \rightarrow_E t[r] \Leftrightarrow l \approx r$$

Handelt sich bei $\rightarrow$ um eine binäre Relation, so bedeuten:

- $\leftarrow$ die inverse
- $\leftrightarrow$ die symmetrische
- $\rightarrow^+$ die transitive
- $\rightarrow^*$ die reflexiv-transitive Hülle

$\leftrightarrow_E^*$ bedeutet also eine Kongruenzrelation.

Ersetzen wir das Vorkommen von s in E(es galt also $E[s]$) durch t, so können wir schreiben $E[t]$.

Ein Term ist in seiner *Normalform*, wenn es keine Ersetzungsregel gibt die ihn in einen Weiteren Term überführt. Hat jeder (Grund)Term genau eine Normalform, so ist die Ersetzung *(Grund)konfluent*. Eine Ersetzung *terminiert* wenn es keine unendliche Folge von Ersetzungsregeln gibt die auf einen Term angewandt werden kann. Ersetzungen die (Grund)konfluent sind und terminieren, heissen Grund*konvergent*.

4.2 Definition der Hülle

Als erstes schauen wir uns noch mal das Konzept einer abstrakten kongruenten Hülle an. Es sei Σ eine Signatur und K eine Menge von Konstanten die von Σ disjunkt ist. E ist eine Menge von Gleichungen über Σ, D ist eine Menge von *D-Regeln*, C ist eine Menge von *C-Regeln*.

- Eine D-Regel ist eine Ersetzungs-Regel der Form $t \to c$. Dabei ist $t \in \mathrm{T}(\Sigma \cup \mathrm{K}) - \mathrm{K}, \mathrm{c} \in \mathrm{K}$.
- Eine C-Regel ist eine Ersetzungs-Regel der Form $c \to d$. Dabei ist $c, d \in K$.

Vertiefen wir dies nun mit einem Beispiel:

- $\Sigma = \{a, b, f\}$
- $E = \{a \approx b, fffa \approx fb\}$
- $K = \{c_0, c_1, c_2, c_3\}$
- $D = \{a \to c_0, b \to c_1, fffa \to c_2, fb \to c_3\}$
- $C = \{c_0 \to c_1, c_3 \to c_3\}$

Die Gleichungen aus E wurden mit Hilfe von Konstanten aus K umgeschrieben. Jedem Term wurde eine Konstante zugeteilt, welche diesen repräsentiert. Diese „Zuteilung“ ergibt die Menge D der *D-Regeln*. Dem Term a wird hier die Konstante c_0 zugeteilt, dem Term fb die Konstante c_3. Die Terme in E werden nun durch die zugeteilten Konstanten ersetzt, was zur Folge die neue Menge $E = \{c_0 \approx c_1, c_2 \approx c_3\}$ hat. Aus dieser Menge werden nun die *C-Regeln* erstellt.

Die Menge $C \cup D$ kann nun als eine Alternative Form von E angesehen werden. Da keine Informationen verloren gehen, kann diese Art der Wiedergabe als „verlustfrei“angesehen werden. Allerdings ist es noch nicht die abstrakte kongruente Hülle, da die Menge nicht grundkonvergent ist. Die Terme sind nämlich noch nicht in ihrer Normalform. Wir können nun die Hülle definieren.

4.2.1 Die formale Definition

Sei Σ eine Signatur, K eine Menge von Konstanten die von Σ disjunkt ist. Ein Ersetzung $R = D \cup C$ bestehend aus D-Regeln und C-Regeln über $\Sigma \cup K$ heißt (abstrakte) kongruente Hülle wenn:

i. jede Konstante $c \in K$ die in Normalform ist einen Term $t \in \mathrm{T}(\Sigma)$ durch R repräsentiert,

ii. R grundkonvergent ist.

und wenn E eine Menge von Grundgleichungen über $\mathrm{T}(\Sigma \cup \mathrm{K})$ ist und des weiteren

iii. R ist so das $\forall s, t \in \mathrm{T}(\Sigma)$ gilt $s \leftrightarrow_E^* t \Leftrightarrow s \rightarrow_R^* \circ \leftarrow_R^* t$

Die Bedingungen bedeuten im einzelnen, daß i keine unnötigen Konstanten erzeugt werden, ii äquivalente Terme von ein und dem gleichen „Repräsentanten“wiedergegeben werden und iii das R eine verlustfreie Erweiterung der Gleichungen von E über $\mathrm{T}(\Sigma)$ ist.

4.3 Die Berechnung der abstrakten kongruenten Hülle

Hier wird eine allgemeine Methode zur Berechnung der kongruenten Hülle vorgestellt. Dazu werden Übergangsregeln vorgestellt, welche auf Mengentripel angewendet werden. Diese Tripel bestehen aus K, einer Menge von Konstanten die die ursprüngliche Signatur erweitern, E der Menge von Gleichungen aus denen die kongruente Hülle berechnet wird, so wie R, der Menge von D-Regeln und C-Regeln die bisher abgeleitet wurden. Die Berechnung führt schrittweise ein Tripel (K_i, E_i, R_i) mit Hilfe einer Übergangsregel in ein neues Tripel $(K_{i+1}, E_{i+1}, R_{i+1})$. Die Berechnung beginnt bei $(\varnothing, E, \varnothing)$ und endet mit $(K, \varnothing, R)$.

Eine kleine Anmerkung: für die Regeln wird eine Ordnung benötigt. Dies ist eine Ordnung $\succ$ über die Terme aus $\mathrm{T}(\Sigma \cup \mathrm{U})$, wobei U eine unendliche Menge von Konstanten ist und es gilt $K \subset U$. Wenn $\succ_U$ eine eine Ordnung auf die Menge U ist , dann ist $\succ$ definiert als: $c \succ d$ wenn $c \succ_U d$ und $t \succ c$ wenn $t \rightarrow c$ eine D-Regel ist. Hier nehmen wir an das die Menge $U = \{c_0, c_1, c_2, \ldots\}$, und $c_i \succ c_j$ wenn $i < j$.

4.3.1 Die Übergangsregeln

Die **Ext**ension ist die wichtigste Regel, sie führt neue Konstanten ein als Namen für Subterme. $t \rightarrow c$ ist eine D-Regel, t ist ein Term aus E und $c \ni \Sigma \cup K$.

$$\frac{(K, E[t], R)}{(K \cup \{c\}, E[c], R \cup \{t \rightarrow c\})}$$

Die **Sim**plification ersetzt Vorkommen von t in E durch c

$$\frac{(K, E[t], R \cup \{t \rightarrow c\})}{(K, E[c], R \cup \{t \rightarrow c\})}$$

Durch wiederholtes Anwenden der Extension und Simplification können alle Gleichungen von E mit Hilfe der
Orientation

$$\frac{(K \cup \{c\}, E \cup \{t \approx c\}, R)}{(K \cup \{c\}, E, R \cup \{t \rightarrow c\})}$$

in eine Ordnung gebracht werden, wenn $t \succ c$.
Die **Del**etion erlaubt uns triviale Gleichungen zu eliminieren

$$\frac{(K, E \cup \{t \approx t\}, R)}{(K, E, R)}$$

Die **Ded**uction erzeugt neue Gleichungen für E

$$\frac{(K, E, R \cup \{t \to c, t \to d\})}{(K, E \cup \{c \approx d\}, R \cup \{t \to d\})}$$

Collapse

$$\frac{(K, E, R \cup \{s[t] \to d, t \to c\})}{(K, E, R \cup \{s[c] \to d, t \to c\})}$$

Und zu letzt **Com**position

$$\frac{(K, E, R \cup \{t \to c, c \to d\})}{(K, E, R \cup \{t \to d, c \to d\})}$$

4.3.2 Ein Beispiel

Wir betrachten die Berechnung der kongruenten Hülle zur folgenden Menge von Gleichungen: $E = \{a \approx b, fffa \approx fb\}$. Auf diese wenden wir nun schrittweise die soeben definierten Regeln an.

i	Konstanten K_i	Gleichungen E_i	Regeln R_i	Übergangsregeln
0	$\varnothing$	E_0	$\varnothing$	–
1	$\{c_0\}$	$\{c_0 \approx b, fffa \approx fb\}$	$\{a \to c_0\}$	EXT
2	$\{c_0\}$	$\{fffa \approx fb\}$	$\{a \to c_0, b \to c_0\}$	ORI
3	$\{c_0\}$	$\{ffc_0 \approx fc_0\}$	$\{a \to c_0, b \to c_0\}$	SIM + SIM
4	$\{c_0, c_1\}$	$\{fc_1 \approx fc_o\}$	$R_3 \cup \{fc_0 \to c_1\}$	EXT
5	$\{c_0, c_1\}$	$\{fc_1 \approx c_1\}$	$R_3 \cup \{fc_0 \to c_1\}$	SIM
6	$\{c_0, c_1\}$	$\{\}$	$R_5 \cup \{fc_1 \to c_1\}$	ORI

Die Menge R_6 ist die von uns gesuchte kongruente Hülle. Die darin enthaltenen C-Regeln entsprechen den Äquivalenzen von Äquivalenzrelationen und wenn es dort zu Widersprüchen kommt, bedeutet das auch eine unerfüllbare Implikation im Gleichungssystem E.

4.4 Berechnungsstrategien

Wie schon erwähnt entstanden verschiedene Lösungsansätze zur mechanischen Programmverifikation welche sich auf die kongruente Hülle stützten. Dabei benutzten Sie auch verschiedene Strategien um diese zu berechnen. Mit Hilfe der Abstraktion zu den Überführungsregeln ist es nun möglich diese direkt zu vergleichen.

4.4.1 Die Methode von Shostak

Shostak's Methode hat die Möglichkeit neue Gleichungen zur Verarbeitung hinzuzufügen, nach dem sie schon einige verarbeitet hat. Das Starttripel lautet $(\varnothing, E_0, \varnothing)$. Sie geht wie folgt vor:

1. Wähle eine Gleichung $s \approx t$ aus E
2. **sim**plifiziere den Term s zu s'
3. benutze **ext**tension bis der s' Term zu einer Konstante(o.B.d.A c) geworden ist
4. führe Schritte 2 und 3 für t durch, bis zur Konstante(o.B.d.A d)
5. ist $c = d$ führe **del**etion aus und fahre fort mit 1, sonst führe **ori**entation aus.
6. führe mit der neuen Regel **Col**apse aus, so wie alle sich daraus ergebenden **Ded**uction-schritte aus.
7. beginne von neuen bei 1.

Als Schema sieht dieser Vorgang wie folgt aus:

$$\mathbf{Shos} = ((\mathbf{Sim}^* \circ \mathbf{Ext}^*)^* \circ (\mathbf{Del} \cup \mathbf{Ori}) \circ (\mathbf{Col} \circ \mathbf{Ded}^*)^*)^*$$

4.4.2 Die Methode von Downey, Sethi und Tarjan

Dieser Algorithmus beginnt mit einem erweiterten Starttripel. Da er sonst mit einem fertigen DAG beginnt müssen wir einige Anpassungen vornehmen. Das Starttripel lautet $(K_1, \varnothing, D_1 \cup C_1)$ und besteht aus D_1 welches den DAG darstellt, so wie C_1 was die Anfangs gegebenen Äquivalenzen enthält. Ebenso wird eine „Leere Übergangsregel“ϵ benötigt.

1. Wenn ein **Col** anwendbar ist, führe ihn durch.
2. Ist danach eine **Ded** anwendbar, führe sie durch oder gehe zu Schritt 1.
3. Wenn keine **Col** mehr möglich sind, wird jede C-Regel-ergebende Gleichung in E **Ext** und entweder **Del** oder **Ori**

Das DST-Schema:

$$\mathbf{DST} = ((\mathbf{Col} \circ (\mathbf{Ded} \cup \{\epsilon\}))^* \circ (\mathbf{Sim}^* \circ (\mathbf{Del} \cup \mathbf{Ori}))^*)^*$$

5 Quellen

- L.Bachmair and A.Tiwari. Abstract Congruence Closure and Specializations. In D.A. McAllester, editor, *Porceedings of the 17th International Conference on Automated Deduction*, volume 1831 of *LNAI*, pages 64-78. Springer Verlag 2000.
- G.Nelson and D.C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM*, 27(2):356-364, 1980.
- David Detlefs, Greg Nelson, James B. Saxe. Simplify: A Theorem Prover for Program Checking. HP Laboratories Palo Alto. HPL-2003-148. July, 16th, 2003. p26-p29.